APPLICATIONS OF MATHEMATICAL PROGRAMMING IN THE DEVELOPMENT OF SOME MODELS FOR PERFORMANCE EVALUATION OF DISTRIBUTED SYSTEMS

A

THESIS

SUBMITTED IN FULFILLMENT

OF THE AWARD OF THE DEGREE OF

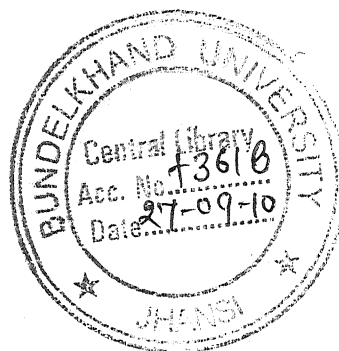DOCTOR OF PHILOSOPHY

IN

MATHS . SC. & COMP. APPL.

By:

POONAM

Under the supervision of:

DR. AVANISH KUMAR
Senior Lecturer
Department of Mathematical Sciences
and Computer Applications

BUNDELKHAND UNIVERSITY
JHANSI-284128, UP (INDIA)

MARCH, 2009

APPLICATIONS OF MATHEMATICAL PROGRAMMING IN THE DEVELOPMENT OF SOME MODELS FOR PERFORMANCE EVALUATION OF DISTRIBUTED SYSTEMS

A

THESIS

SUBMITTED IN FULFILLMENT

OF THE AWARD OF THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

MATHS . SC. & COMP. APPL.

By:

POONAM

Under the supervision of:

DR. AVANISH KUMAR
Senior Lecturer
Department of Mathematical Sciences
and Computer Applications

BUNDELKHAND UNIVERSITY
JHANSI-284128, UP (INDIA)

MARCH, 2009

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **"APPLICATIONS OF MATHEMATICAL PROGRAMMING IN THE DEVELOPMENT OF SOME MODELS FOR PERFORMANCE EVALUATION OF DISTRIBUTED SYSTEMS"** in fulfillment of the requirement for the award of the Degree of **Doctor of Philosophy** in ~~**Mathematics**~~, *Maths. Sc & Comp. Appl.* submitted in the Department of Mathematical Sciences and Computer Applications, Bundelkhand University, Jhansi is an authentic record of my research work, under the kind supervision of **Dr. Avanish Kumar**, Senior Lecturer, Department of Mathematical Sciences and Computer Applications, Bundelkhand University, Jhansi.

It is also certify that I have not submitted the matter embodied in this thesis for the award of any other degree.
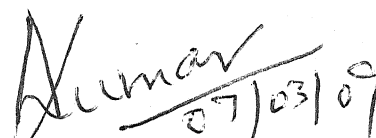
(Poonam)

Dated: 7-3-09

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge. It is also certify that she has put a minimum number of attendance i. e. 200 days in the department as required by the university.

Supervisor

27/03/09

(Dr. Avanish Kumar)
Senior Lecturer
Department of Mathematical Sciences
and Computer Applications
Bundelkhand University
Jhansi-284128, UP(INDIA)

ACKNOWLEDGEMENT

(Poonam)

Dated: 7 – 3 – 09

Place: Jhansi

CONTENTS

INTRODUCTION

1.1. INTRODUCTION TO MATHEMATICAL PROGRAMMING

1.1.1. An Overview

Mathematical Programming gained impotence with the advent of *"Operation Research"* with comprises of applications of mathematical analysis to managerial, Scientific and computational problems. The operation research or management science approach to solving a problem is based on the scientific method. Sir Francis Bacon first started it in 1960. Bacon felt that problem inquiry should consist of the following four steps:

- Observation and problem description

- Hypothesis statement

- Model development and test

- Model analysis

During the last 1930s in England and early 1940s in United States, the use of scientific methods was extensively made to analyze optimization problems. World War II challenged both countries at this time to develop optimal solution for allocation, transportation and multi-variable type problems. The modeling techniques studied by

operations researchers in 1940s and 1950s usually required algebra or calculus for solution purposes. The term *"Mathematical programming"* was used then, and is still used today, to describe the structuring of mathematical symbols into a model or program. Mathematical programming problems first arose in the field of Economics where allocation problems had been a subject of deep interest.

During World War II, a group of researchers sought to solve allocation type problems for the United States Air Force. One of the members of this group formulated and devised a solution procedure in 1947 for linear programming type problems. This solution procedure, called the simplex method, marked the beginning of the field of study called Mathematical Programming. Mathematical Programming is useful for finding an optimal solution to a complex problem involving many interrelated variables. It also simplifies exposition of many network problems and helps in finding optimal results which management must implement to realize its objectives.

In 1972, [TURB72], surveyed a large number of United States Corporations on their use of operation research activities. One of his conclusions was that individuals within organization were more often viewing decisions making situations as a management science type problem requiring some types of mathematical modeling, than ever before observed in the past. Markland and Newett [MARK1972] voiced concern in their study on the misuse of mathematical programming and future problems that can be caused by the ineffective use of management science practitioners who fail to consider implementation as a part of every study undertaken in management science. Some

surveys [RADN1973, Gray1973, FABO 1976] sought specific information on the use of mathematical programming techniques. One of these surveys conducted by Fabozzi and Valaente [FABO1976] examined the use of mathematical programming methodology and where in the organization it was used. A similar study was conducted by Ledbetter and Cox [LEDB77]. They found that many organizations were using the mathematical programming techniques as reported by Fabozzi and Valaente as well as other management science methodologies. Mathematical Programming techniques have increased in number over the years. An important area of development of mathematical programming concerns with the use of computer in management science. More and more computer software are being developed every day making the computational aspects of mathematical programming less complex. Time-sharing and interactive computer systems are also influencing a de-emphasis on computational experience and more on problem formulation. During the last decade, mathematical programming approaches have gained immense importance for solving task allocation and load balancing problems in Distributed Processing Environment (DPE) [CHU1980, MA1982, SINC1987, and REID1994].

1.1.2. Mathematical Programming Problems

During the last decade, mathematical programming techniques have been recognize as the most powerful methods for modelling and analyzing several kind of problems. Many research problems are formulated in the form of mathematical models, which describe the quantitative features of all types of problems. Mathematical programming techniques are used for the formulation and solution of research problems

by systematic planning of various activities. The basic problem in mathematical programming is to find the unknown values of some variables, which will optimize the value of the objective function subject to a set of constraints. Most of the mathematical programming problems can be formulated in the following general form [KWAK 1987]:

$$Maximize \ (or \ Minimize) \ Z = \sum_{j=1}^{n} c_j x_j$$

$$subject \ to \ \sum_{j=1}^{n} a_{ij} x_j <=> b_i \ (for \ i = 1,2,........,m)$$

$$and \ x_j \geq 0 \ (for \ j = 1,2,........, n)$$

Where,

Z = value of the objective function which measures the effectiveness of the decision choice,

x_j = unknown variables that are subject to the control of the decision maker,

c_j = unit profit contribution of an output or cost of an input which is known,

a_{ij} = production (or technical) coefficients that are known, and

b_i = available resources in limited supply.

Production of goods, in any organization, requires productive resources, which are in short supply in the real world, and they are, therefore, restricted resources. These restricted resources are expressed as equalities or inequalities in mathematical programming models.

1.2. INTRODUCTION TO COMMUNICATION

1.2.1. An Overview

The reasons for studying data communications can be summed up in the occupational history of the United States. In the 1800s they were an agricultural society dominated by farmers. By the 1900s they had moved into an industrial society dominated by labor and management. Now, as they approach the twenty-first century, they clearly have moved into the information society, which do computers, data communications, and highly skilled individuals who use brainpower instead of physical power dominate. The industrial society has reached its zenith, and the communication/computer era, started in mid 1950s, which is dubbed the information society, is advancing rapidly.

In an information society dominated by computers and communications, value is increased by knowledge as well as by the speed of movement of that knowledge. This new information economy will completely destroy Ricardo's labor theory of value, because, in such a society, what increases value is not the labor of individuals, but information. The main stream of the information age is a communication network. The value of a high-speed data communication network that transmits knowledge/information is that it brings the message sender and the message receiver closer together in time. For example, in the 1800s it might have taken several weeks for specific information to reach the United States from England. By the 1900s it could be transmitted within an hour. Today, with modern data communication systems, it can be transmitted within seconds. Finally, the transition from an industrial to an information society means that we have to

learn many new technologically based skills. The study of data communications has become a basic tool that can be used throughout our lifetime. We incorporate our knowledge of data communications into several careers such as circuit designer, programmer, business system application developer, communication specialist, and business manager.

1.2.2. Evolution

Today we take data communications for granted, but it was early pioneers like Samuel Morse, Alexander Graham Bell, and Thomas Edison who developed the basic electrical and electronic systems that ultimately became today's voice and data communication networks. When the telephone arrived, it became the accepted communication device that everyone wanted. Several technological enhancements were made in telephonic technology from 1837 to 1951 and it was in 1951 that the first direct long distance customer dialing began. The first international satellite telephone call was sent over the Telstar satellite in 1962. In 1963, touch-tone telephones began to be marketed. Their push buttons were easier to use than rotary dials, and they became quite popular. By 1965, there was widespread introduction of commercial international telephone service by satellite. Picturefone service, which allows users to see as well as talk with one another, began operating in 1969. All through the 1970s there were many arguments and court cases regarding the monopolistic position that A T &T held over other companies that wanted to offer communication services. The litigation led to the litigation led to the divestiture of AT&T on January 1, 1984. During 1983-84 the newer cellular telephone networks supplanted traditional radio telephone-type calls. Integrated

Services Digital Networks began serving the public in 1986. These networks allow the simultaneous transmission of voice, data, and video images [FITZ1988]. By 1987 there was considerable competition in both the voice and data communication markets as a number of independent companies began to sell communication services in a manner similar to that of automobile marketing. And now we have smaller and less expensive portable telephones to carry around everywhere.

India is a resource rich fast developing country where telecommunication has passed from the stage of convenience to essentiality. Computers have recently entered in a big way in Indian society. As on March 31 1986, it was estimated that the country had about 3050 super, mainframes and minis and 7000 micros and personal computers. The number of microcomputers and personal computers is growing fast. Today it has crossed the limit of one million. It was thus thought to have the computer facilities. The idea of computer networking has been widely accepted in government organizations. The NIC has been doing this for planning purposes to Govt. of India. NICNET computer and communication facilities have provided nation-wide links to make the computing resources available at the places from where the information emanates [AGAR1995]. The Indian society has now clearly realized the importance of communications, which may be in its various forms including telephones, memorandums, telex, mail, reports etc. Attempts have been made by the organizations like telecommunications research center to produce prototype model of several important communication systems. Electronic mail, facsimile equipment, modems, teletext voice mail etc. have been developed, integration of computer communication with special digital switches, ISDN, has also

been tried. This has put India in the forefront of telecommunication services. The Govt. of India has realized that without a better and secures telecom system the fruits of development can never be fairly distributed, be it in education, economics, rural development or international trade and banking. The development of efficient communication systems has opened new vistas in electronic industries. The OSI technology is coming up fast and Fiber Optics technology has also entered into the scene. Optical fiber computer communication networks [Fibre LANs] are now being developed in India.

1.3.   DISTRIBUTED SYSTEM

Distributed processing plays an important role in large data base installations where processing load is distributed for organizational efficiencies. Banking system, travel agency systems, and power control systems are few examples of distributed processing environment. The application that exhibits parallelism, involving enormous repetitive processing, and/ or requiring extremely fast processing in a real-time environment demand distributed processing systems. To name a few such type of application are signal processing, meteorology, image analysis, cryptography, nuclear reactor control, sonar & radar surveillance, simulation of VLSI circuits, and industrial process monitoring. The on-set of the microprocessor technology has made the Distributed Processing System [DPS] economically viable and attractive for many applications of computer. However, many problem areas in DPS are still in their primitive development stages. DPS are increasingly drawing attention, yet have a meaning that is not understood. The term "DPS" is used to describe whenever there are

several computers interconnected in some fashion so that a program or procedure running system terms with multiple processors. However, the term has different meanings to different systems because processors can be interconnected in many ways for various reasons. In the most general form, the word distribution implies that the processors are in geographically separate locations. Occasionally, the term is also applied to an operation using multiple mini-computers, which are not hardware, connected with each other and are connected through satellite. A user-oriented definition [BHUT1994, SITA1995] of distributed computing is "Multiple Computers, utilized cooperatively to solve problems". While addressing task allocation issues in a DPS, the three important aspects need consideration:

(i)   Multiplicity of resources:

There are a number of resources, in particular, processors. Homogeneity of physical resources is not essential. A system may have processors with identical characteristics and capabilities.

(ii)   Dispersion:

The resources in the system are physically or logically distributed. AU the processors are independent and tied together by communication links. The communication links provide means for transferring information between the processors.

(iii)   Control:

All the processors in the system are autonomous and there is no master and slave relation among the processors. For the user, the collection of processors should be invisible, the multiple processor system should appear as virtual uniprocessor, and users need not know on which machine their programs are running and where their files are stored.

(iv)   Transparency:

All the processors in the system are autonomous and there is no master-slave relationship among them. The fact there are several co-operating processors in the system must be invisible (transparent) to the user and a single image id presented to the user. The system should appear as a uniprocessor system and users need not know on which machine their programs are executing and where the data or files stored.

(v)   Flexibility:

Flexibility is an important aspect of a distributed system. It means that should be easy to incorporate changes in a user transparent manner or with minimum interruption to the user. Further, in every system, new functionalities are to be added from time to time to make it more powerful and easy to use. Therefore, It should be easy to add new services to the system.

(vi)   Performance:

The performance of the distributed system must be at least as good as a centralize system. That is when a particular application is run on distributed system, Its overall performance should be better than or at least equal to that of running the same application on a single processors system.

(vii)   Scalability:

Scalability refers to the capability of a system to adapt to increased service load. A distributed system must able to cope with the growth of nodes and users in the system.

(viii)   Security:

Security must be provided in the system to prevent destruction and unauthorized access so that user can trust on the system and rely on it.

1.3.1. Distributed Real Time Systems

The on-set of the "Microprocessor Technology" has made the Distributed Real-Time Processing System (DTRS) economically viable and attractive for many applications of computer. However, many problem areas in Distributed Real-Time Processing System are still in their primitive development stages. Distributed Real-Time Processing System are increasingly drawing attention, yet have a meaning that is not understood. The term "Distributed Real-Time Processing System" is described whenever several computers interconnected in some fashion such that a program or procedure

utilizes this distributed but combined power and gets executed in real time. The term has different meanings with regard to different systems, because processors can be interconnected in many ways for various reasons. In its most general form, the word distribution implies that the processors are fixed in geographically separated locations. Occasionally, the term is also applied to an operating environment using multiple mini-computers not connected with each other with the help of physical communication lines but are connected through satellite.

1.3.2. Machine Size Vs Instruction Execution

Distributed system provides much faster execution by facilitating parallel execution of tasks. A major driving force towards distributed processing is the cost of small processors. Until the spread of mini computers in the early 1970s, a commonly accepted rule was Grosch's Law, which said, "The cost per machine instruction executed is inversely proportional to the square of the size of the machine ". Grosch's Law became questionable in the 1970s. Even some people suggested that it had been reversed because the cost per instruction on some mini-computers was lower than on large computers and on microprocessors was lower than mini-computers. The reason is related to the use of Very Large Scale Integrated circuits, which can be mass-produced economically. Their development cycle is much shorter than that of large machines.

1.3.3. The Logical Vs Physical Design

The implementation of a distributed system depends both on logical and physical premises. The logical functions center on the procedural design for channeling the flow

of data and controlling the physical facility throughput of the projected system configuration. The object' of the physical functions is the design of the hardware and hard-software [firmware] devices to provide a specific level of capability. Functions will be distributed both to machines in the computer center and to many machines at user locations. This trend of the distribution of processing is continuing because software usage of machine instructions per second is growing much faster than the development of higher speed machines. The distributed system is motivated by the need for cost reduction in tasks executing. The chronological order of development of the distributed system requiring the logical and the physical design phase is depicted in following figure:

Distributed
Processing System

Logical Distributed
Processing System

Physical Distributed
Processing System

Model Development

Algorithm Design

Software Design

Implementation

Integration
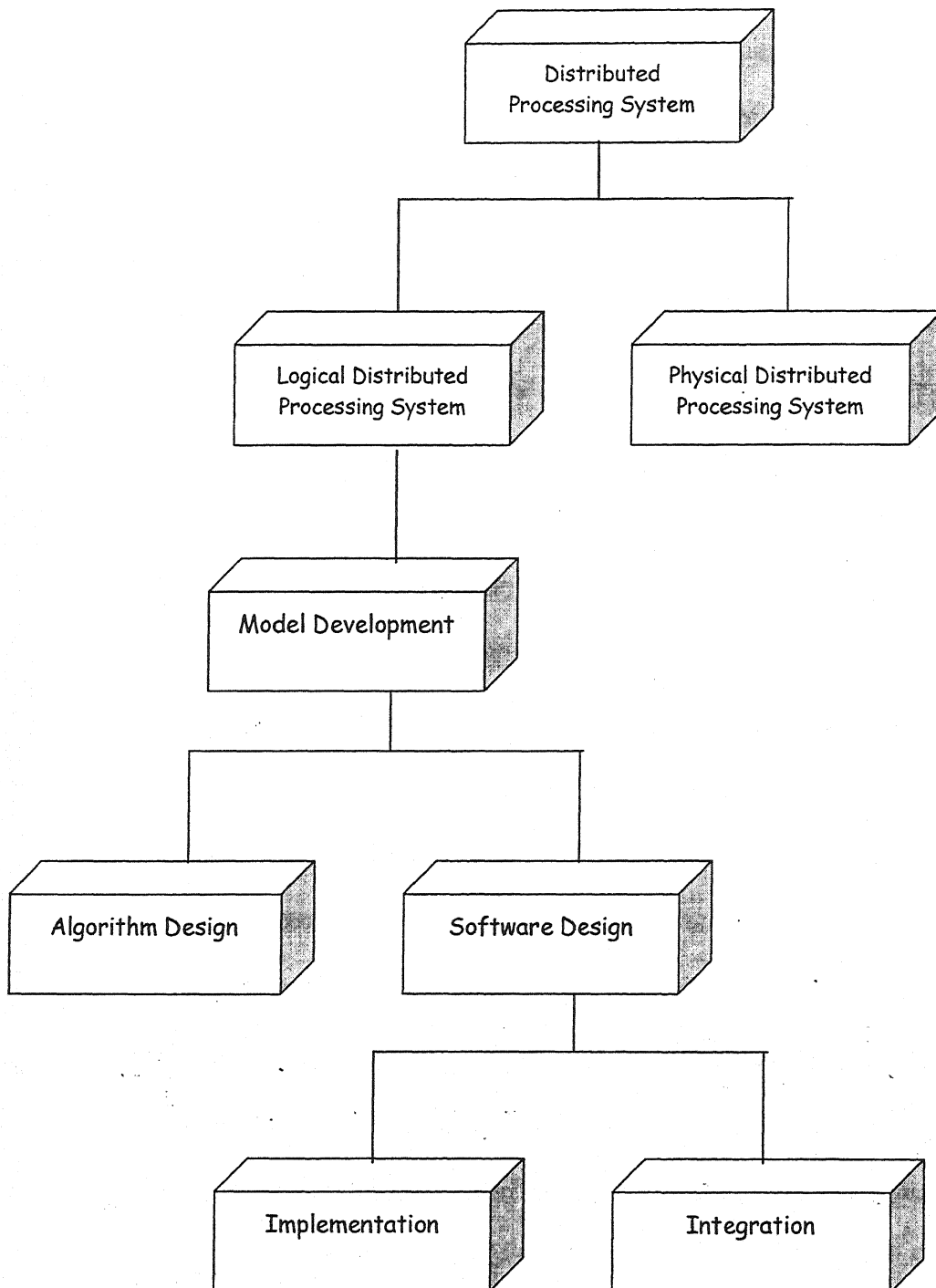
1.3.4. Distributed Vs Parallel Computing

In some computer networks, the control mechanism is mostly centralized. In others, they are mostly distributed. Where purely centralized control exists, loss of the center puts the entire network out of action. With the distributed control any portion of network can be destroyed and the rest will continue to function. Although computation speed has increased several folds over the past three decades of computing, the user demand for faster speeds is growing at a much faster rate. One of the approaches to meet the growing demand of faster computation is to use parallel processing. Parallel computers, which emphasize parallel processing, may be employed. Parallel computers are available with different architectures [FORT1985, STON1987, BOKH1988] and divided in to three classes: array computers, pipeline computers and multiprocessor systems. To name a few are IBM 3081, Denelcor HEP, Cray x-MP and PARAM systems. All the parallel computers described above are centralized computing items and all hardware and software resources are housed at the same place.

1.3.5. Types of Distributed Processing Systems

1.3.5. 1. Horizontal Vs Vertical Distribution

By Vertical distribution we mean that there is a hierarchy of processors, shown in following figure [MART 1988]. The transaction may enter and leave the computer system at the lowest level. The lowest level may be able to process the transaction or may execute certain functions and pass it up to the next level. Some, or all, transactions eventually reach the highest level, which will probably have access to on-line files or

databases. The computers at the lowest level can be networked together, if data sharing is required.
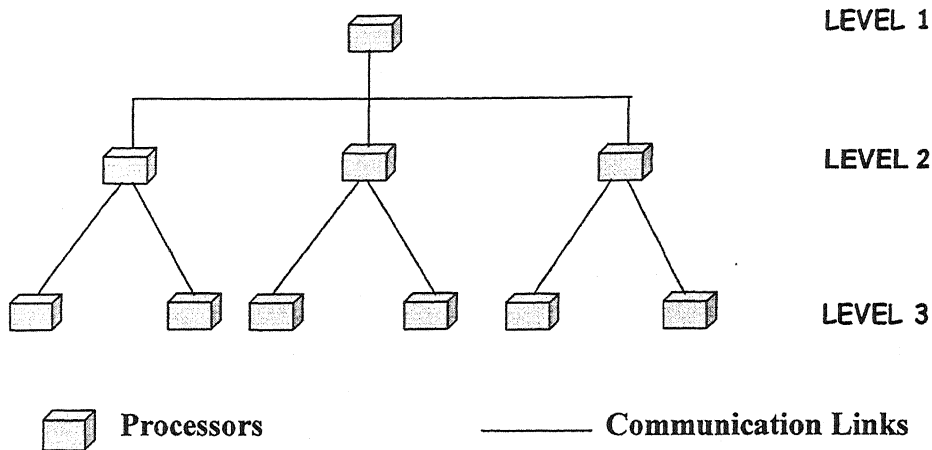
Vertical Distributions

A horizontal distribution implies that the distributed processors do not differ in rank. They are of equal status-peers and referred as peer-coupled systems. A transaction may use only one processor, although there are multiple processors available. On some peer-coupled systems a transaction may pass from one system to another causing different set of files to be updated as depicted in the following figure [MART 1988].
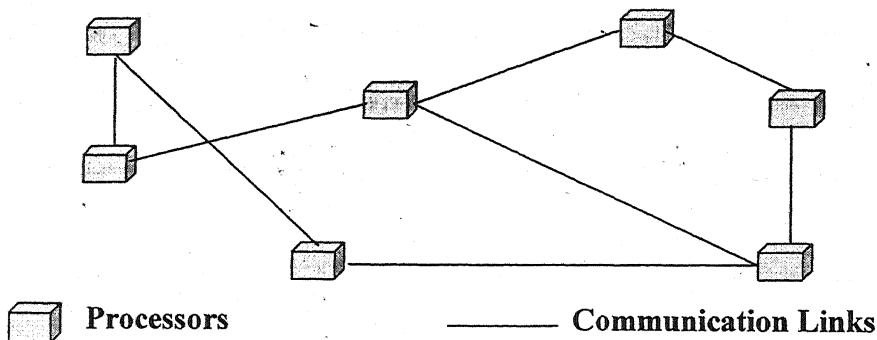
Horizontal Distributions

1.3.5.2. Functional Distribution Vs System Distribution

In some distributed systems, usually vertical systems, functions are distributed, but not the capability to fully process entire transactions. The lower-level machines may be intelligent terminals or intelligent controllers in which processors are used for functions such as message editing, screen formatting, data collection dialogue with terminal operators, security, or message compaction concentration. They do not complete the processing of entire transactions. This distribution is referred as functional distribution and is contrasted with system distribution in which the lower-level machines are systems their own right, processing their own transactions, and occasionally passing transactions or data up the hierarchy to higher level machines. In a system distribution environment the lower machines may be entirely different from, and incompatible with, the higher machines. In a function distribution environment, close cooperation between the lower-level and higher-level machines is vital. Overall system standards are necessary to govern what functions are distributed and exactly how the lower and higher machines form part common system architecture with appropriately integrated control mechanisms and software.

1.3.6. Features of Distributed System

There are a variety of reasons for building of any distributed systems, some of them are as follows:

(i) Computational Speedup:

Computational speedup can be achieved if a particular computation can be partitioned into a number of sub-computations that can run concurrently. A distributed

system may allow user to distribute the computation among the various sites – to run that computation concurrently. In addition, if a particular site is currently overloaded with the jobs, some of them may be moved to other, lightly loaded, sites.

(ii) Fault Tolerance:

One of the real attractions of distributed processing is the resulting high fault tolerance. If a communication link or a site fails in a distributed system, the remaining sites can potentially continue operating. However, this may reduce the overall throughput of the system.

(iii) Increased Throughput:

The throughput of the system is expected to increase by distributing the total workload to the various service stations. As in a distributed system there are a number of processing elements, one would hope to get more work done in shorter period of time.

(iv) Communication:

In the distributed system it is quite often the programs running at different sites need to exchange data with one another. When a number of site are connected to one another by a communication network, the processes at different sites have the opportunity to exchange the information.

(v) Resource Sharing:

If a number of sites are connected to one another, then a user at one site may be able to use the resources available at another. In general, resource sharing in a distributed

system provides mechanisms for sharing files at remote sites, processing information in a distributed database, printing files at remote sites, utilizing specialized hardware devices, and performing other operations. Distributed processing plays an important role in large data base installations where processing load is distributed for organizational efficiencies. Banking system, travel agency systems, and power control systems are few examples of distributed processing environment. The application that exhibits parallelism, involving enormous repetitive processing, and/ or requiring extremely fast processing in a real-time environment demand distributed systems. To name a few such type of application are signal processing, meteorology, image analysis, cryptography, nuclear reactor control, sonar & radar surveillance, simulation of VLSI circuits, and industrial process monitoring.

1.4   TASK ALLOCATION

The task allocation in a distributed system finds extensive applications in the faculties where large amount of data is to be processed in relatively short period of time, or where real-time computations are required. Meteorology, Cryptography, Image Analysis, Signal Processing, Solar and Radar Surveillance, Simulation of, VLSI circuits and Industrial process monitoring are areas of such applications. These applications require not only very fast computation speeds but also different strategies involving distributed task allocation systems. In such applications the quality of the output is proportional to the amount of real-time computations. To meet such challenging computing requirements at electrifying speeds some efficient task allocation strategies are required for proper utilization of distributed system under the constraints of memory

capacity available at each processor and time constraints in executing task. The advent of VLSI technology resulting in low cost microprocessor has made distributed system an economic reality in today's computing environment. The modularity, flexibility and reliability of distributed system make it attractive to much type of users, and several distributed systems have been designed and implemented in recent years. The first step in the distributed software engineering is to partition application program into a set of smaller independent tasks and allocates them to different processors. To enable the increasing variety of computers to communicate with one another, there must be rigorously defined protocols as to (a) how the control message and data message are exchanged in the distributed system and (b) how to control the communication process along with the protocol definition.

The format of the control messages, the headers and the trailers of data messages are likewise rigorously defined. Protocol becomes quite complex, as it is desirable that there should be a widely accepted standard so that all types of machines can inter communicate. However, many problematic areas in the distributed systems are still in their Primitive development stage. Some major problems that present the widespread use distributed systems are: (a) the degradation in system throughput caused by the saturation effect, (b) the difficulty in evenly utilizing each processor in the distributed systems, (c) a large gap between the engineering application requirements and existing distributed network architecture and (d) the difficulty in verifying task allocation resulted from any allocating model.

Distributed systems have been so complex that intuition alone is not sufficient to predict their performance. Therefore, mathematical modeling plays an important role for predicting the performance of the distributed systems. Mathematical models of system performance range from relatively simple ones, whose solution can be obtained analytically, to the complex ones, which require simulation. Assigning tasks to processors is called task allocation, which involves the allocation of tasks to processors in such a way that some effectiveness measures are optimized. If the effectiveness measure can be represented as a linear function of several variables subjected to a number of linear constraints involving these variables, then the task allocation is classified as a Linear Programming Problem. Likewise, for the processor, which can perform anyone the several tasks, possibly the difference of execution, and the effectiveness measure is the total execution cost to perform all tasks when one and only one task is allocated to each processor. In such cases, task allocation is classified as an assignment problem. Assigning "m" tasks to "n" processors, through exhaustive enumeration, results in n^m possible ways. A general structure of task allocation in a distributed system is shown in following figure:

```
┌─────────────────────────────┐
│ ❑ Interprocessor Distance Attribute │
│   Bound for each Processor   │
└─────────────────────────────┘

┌──────────────────┐
│ Distributed      │      ┌────────────────────────────┐
│ Processor        │      │ ❑ Simulate the Application Task and │
│                  │      │   Network to verify the Allocation │
│ Application System│     └────────────────────────────┘
│ Modulator        │
│                  │      ┌────────────────────────────┐
│ Application      │      │ ❑ Minimum Cost             │
│ Model            │      │ ❑ Load Balancing           │
│                  │      │ ❑ Application Requirements  │
│ Tasks            │      └────────────────────────────┘
│ Processor        │
│                  │      ┌────────────────────────────┐
│                  │      │ ❑ Coupling Factor of Tasks │
│                  │      │ ❑ Attributes of Tasks      │
│                  │      │ ❑ Size of Attributes       │
└──────────────────┘      └────────────────────────────┘
```
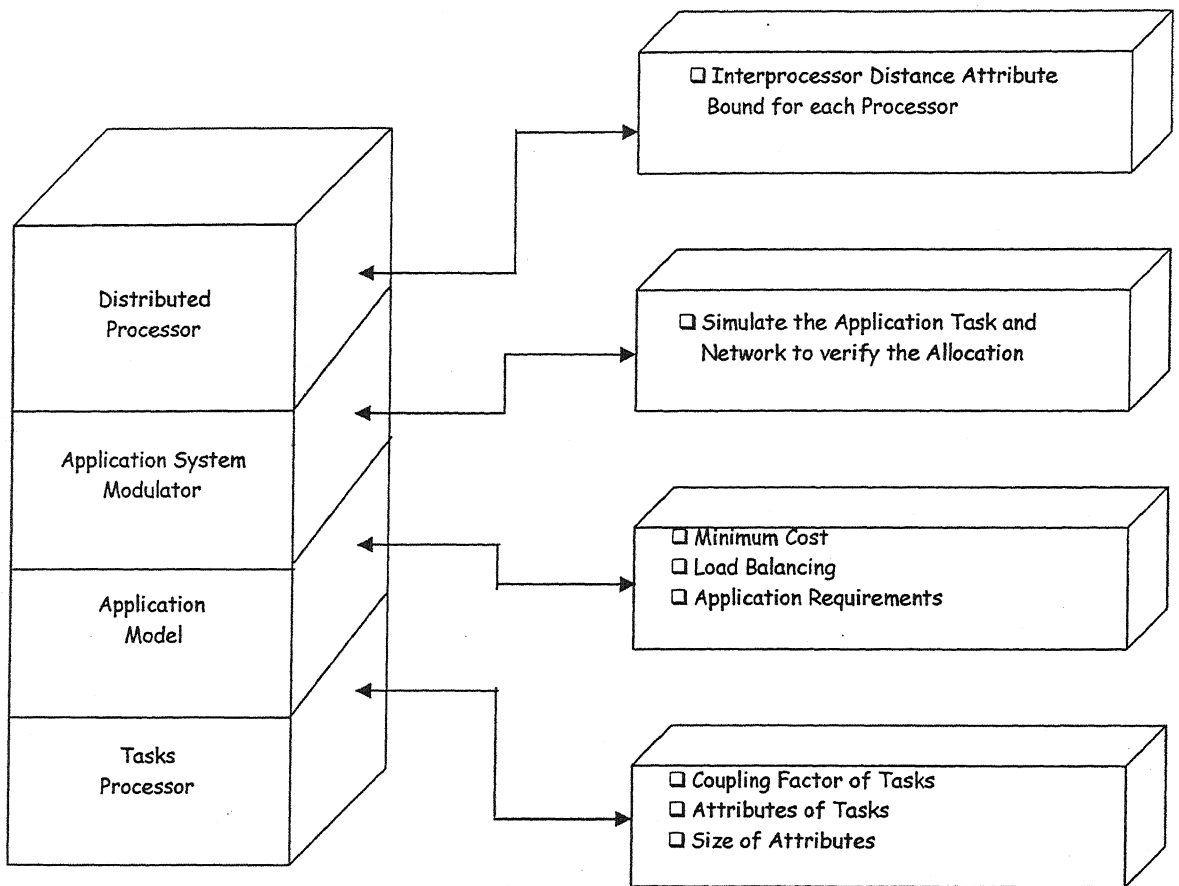
General Structure of Tasks Allocation

Shatz and Wang [SHAT 1987] studied that the problem of choosing an optimal allocation from all assignments is exponentially complex. An efficient task Allocation policy should avoid excessive Inter Processor Communication [IPC] and exploit the specific efficiencies of the processors and in case of a system having similar processor, the tasks or modules should be distributed as evenly as possible. The bottleneck in IPC is to provide linear speed up solutions with the increase in number of processors as

suggested by Chu et al and Lint et al. [CHU 1980, LINT 1981]. The strategies of task allocation on a parallel and distributed system may be done in any of the following ways:

(i)   Static Allocation:

In static allocation when a task is assigned to processor, it remains there while the characteristic of the computation change, a new assignment must be computed. The phrase "characteristics of the computation" means the ratios of the times that a program spends in different parts of the program. Thus in a static allocation, one is interested in finding the assignment pattern that holds for the life time of a program, and result in the optimum value of the measure of effectiveness.

(ii)   Dynamic Allocation:

In order to make the best use of resources in a distributed system, it is essential to reassign modules or tasks dynamically during program execution, so as to the advantage of changes in the local reference patterns of the program [MANI 1998]. Although the dynamic allocation has potential performance advantages, Static allocation is easier to realize and less complex to operate.

1.4.1   Task Allocation Problem

Consider a set of "n" processors P= {p_1, p_2, p_3,........., p_n}, interconnected by communication links and a set of "m" executable tasks T= {t_1, t_2, t_3,........., t_m}. The allocation of each "m" tasks to "n" available processors such that an objective cost function is minimized subject to the certain resource limitations and constraints imposed

by the application or environment. An assignment of tasks to processors is defined by a function f, from the set of tasks T to the set of processors P, f: T→ P. Then the total cost of an allocation x can be expressed as follows:

$$Cost(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left[e_{ij} x_{ij} + \sum_{k<i} \sum_{l<j} c_{ik} d_{jl} x_{ij} x_{kl} \right]$$

where,

$$x_{ij} = \begin{cases} 1, & \text{if } i^{th} \text{ task is assigned to } j^{th} \text{ processor} \\ 0, & \text{otherwise} \end{cases}$$

e_{ij}	=	cost of executing task t_i on processor p_j
c_{ik}	=	Communication between tasks t_i and t_k
d_{jl}	=	distance between processors p_j and p_l

1.5. SCHEDULING POLICIES

The scheduling policies in distributed systems have two categories such as job scheduling and task scheduling. Job scheduling allocates independent jobs to different processors to optimize system performance. Task allocation scheduling requires assignment of multiple interdependent tasks or modules of a single allocation program to minimize job completion time. The decision of scheduling polices is required at several levels within distributed system as shown in figure mentioned below.
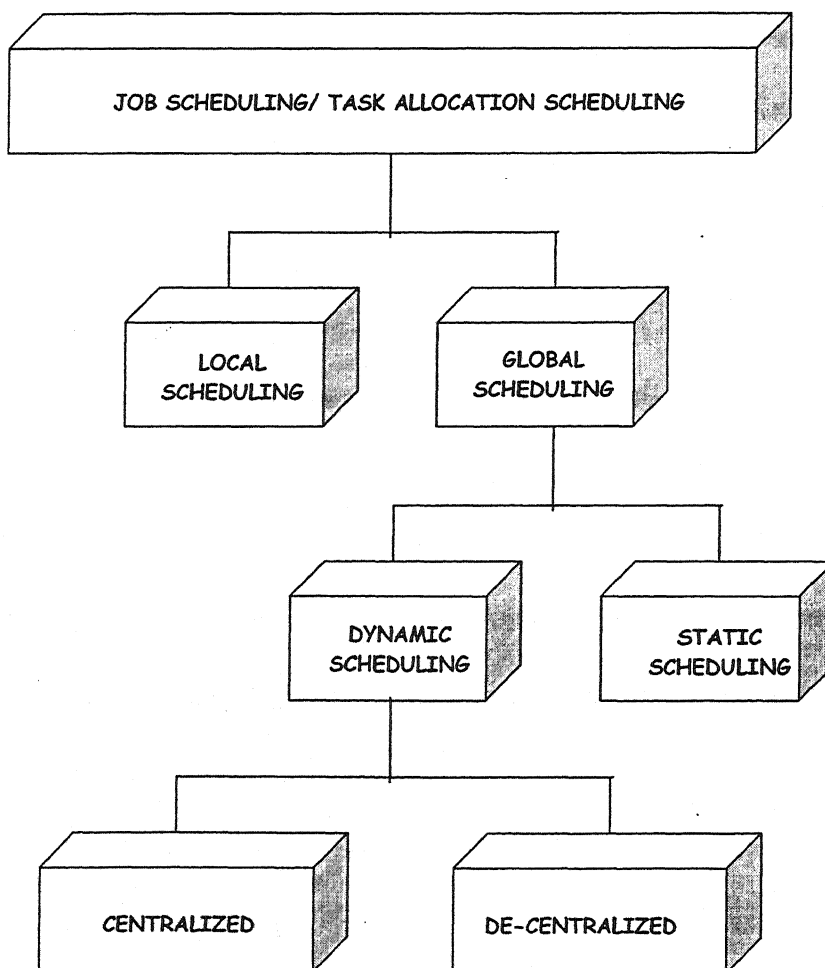
Taxonomy of Scheduling

At higher level the decision about local or global scheduling might be desired to make. The local scheduling policies decide about the allocation of a task or a job to the time slots for a single processor [BUCK1979, CASA1988]. The global scheduling relates to the problem of deciding where a job or a task [ROTI1994] runs. In the next lower level is the choice between static and dynamic in a global scheduling. Static scheduling is prior assignments tasks to the processors where the allocation does not change during the lifetime of tasks. While in dynamic scheduling the allocation decision is made during execution of tasks. Further there are two type of dynamic scheduling: centralized and decentralized. If there is a single decision point, the dynamic scheduling is known as centralized and if the decision-making is spread throughout the system it is known as de-centralized.

1.6. RELIABILITY

The reliability theory has grown into an engineering science in its own right. Much of the initial theory, engineering, and management techniques centered about hardware, however, human and procedural elements of a system were often included. Since its beginning, following World War II, and also then in the late 1960s the term software reliability has become popular, and now reliability theory refers to both software and hardware reliability [SHOO 1983, AMAR 1998, DENS 1998]. Human reliability analysis is a method by which the probability of a system required human action, task, or job will be completed successfully within the required time period and that no extraneous human actions detrimental to system performance will be performed. Results of human reliability analysis's are often used as inputs to probabilistic risk assessments, which

analyze the reliability of entire systems by decomposing the system into its constituent components, including hardware, software, and human operators. Reliability allocation is the process of specifying a level of reliability for each subsystem or module in a system so as to achieve a system reliability objective. Reliability optimization has attracted many researchers since 1960 due to reliability critical importance in various kinds of systems. To maximize system reliability, the following options can be considered:

- Enhancement of component reliability,

- Provision of redundant components in parallel, and,

- Reassignment of interchangeable components.

The diversity of system structures, resource constraints, and options for reliability improvement has led to the construction and analysis of several optimization models.

1.7. ORGANIZATION OF THE THESIS

The First chapter of this thesis entitled "APPLICATIONS OF MATHEMATICAL PROGRAMMING IN THE DEVELOPMENT OF SOME MODELS FOR PERFORMANCE EVALUATION OF DISTRIBUTED SYSTEMS" is devoted to the introductory background of the topic in Mathematical programming, communication, distributed systems, task allocation and other issues directly related to the work. A brief overview of the communication and distributed system has been given to understand the basics of it. Task allocation problem in these systems, is discussed and also some of the related concept has been included. Finally, the organization of the thesis,

include the chapter wise brief summary of the present work, has been mentioned in this chapter.

Task assignment for any distributed system is a most interesting and demandable research problem. Various methodologies and techniques are available in the literature to solve such problems. As it is the primary for researchers to know about related methodology and techniques. The Second Chapter of this thesis is a collection of all such research work, which is available in the literature, and directly-indirectly correlates to our work. We have considered the Decomposition approach, Genetic algorithmic approach, Integer Programming approaches, Dynamic Programming, Heuristic approaches, Network flow & Partitioning algorithm, Problem Reduction method, Petri Net modeling & Reliability evaluation and Shortest Path algorithmic method. And also precedence constraints, particular architecture consideration, reliability evaluation has mentioned.

The Third Chapter of the thesis is the important and main, as it include all five research models. The general motivation for the present research has been written briefly. The Objective, Technique, Algorithm, Implementation and Conclusion of each model are mentioned separately. The details of each model is as mentioned below:

The title of the Model – I is the 'reliability based algorithm for performance enhancement of the distributed systems'. The model is developed with the objective of maximizing the overall processing reliability. Here we have considered the processing reliability and communication amongst the tasks. The results shows that the optimal assignments with optimal reliability. The present model has also compared with Richard

et al [RICH 1982] and found that the developed model is better than the existing one at front of time complexity. Results are shown graphically as well as in tabular form. This study is capable to deal all such real life situations, where the tasks are more than the number of processors. The developed algorithm is programmed and several sets of data have been tested to verify the effectiveness of the algorithm.

The title of the Model – II is the 'task allocation through reliability index for distributed system'. In this model a set of m tasks are assigned to a set of n processor. Inter task communication cost of the task have been considered along with the processor reliability and processing cost. Total processing reliability and cost have been obtained and then reliability function is defined. Model has also been compared with Zahedi et al [ZAHE 19991] and Kumar et al [KUMA 1999]. Results are shown in tabular form and also graphically. The developed algorithm is programmed and several sets of data have been tested to verify the effectiveness of the algorithm.

The title of the Model – III is 'an efficient algorithm for tasks allocation to the distributed systems', with the objective to obtain the optimal assignments in such a way that the execution cost, execution reliability, communication cost and communication reliability has to be optimized. A technique has been devised to fulfill the said objective. The results are shown graphically and also mentioned in the tabular form. To test the performance of the algorithm this model is compared with the existing ones and found that it is a good approach. This study is capable to deal all such real life situations, where the tasks are more than the number of processors. The developed algorithm is

programmed in C++ and several sets of data have been tested to verify the effectiveness of the algorithm.

The title of the Model – IV is the 'communication reduction based algorithm for optimal assignment in distributed systems'. Here we have suggested a way to assigning tasks which have maximum communication. The developed algorithm provides the optimal cost of assignments. The developed model has been compared with Yadav et al [YADA1995] algorithm. This study is capable to deal all such real life situations, where the tasks are more than the number of processors. The developed algorithm is programmed and several sets of data have been tested to verify the effectiveness of the algorithm.

The title of the Model – V is the 'optimizing the reliability of distributed systems', it defines the executing reliability and communication reliability function to obtain the optimal reliability of distributed system. Here we have obtained the processor wise execution reliability. The communication reliability and total reliability of the distributed system has mentioned in the conclusion part of the model. This study is capable to deal all such real life situations, where the tasks are more than the number of processors. The developed algorithm is programmed and several sets of data have been tested to verify the effectiveness of the algorithm.

The future scope of the research has been given in the chapter four of this thesis. During the period of this research work some of the research papers have been presented in seminar and conferences and some of research papers have been sent for the

publication in journals. A combined list of such papers has been mentioned in this chapter. During this research work we have gone through large number of research papers, books, monographs, Ph. D. thesis etc. so that in the last of the thesis a list in alphabetic order of such research material has been attached.

SURVEY

2.1 AN OVERVIEW

Distributed systems are characterized by resource multiplicity and system transparency. Every distributed system consists of a number of resources interconnected by a network. Besides providing communication facilities, the network facilitates resource sharing by migrating a local process and executing it at a remote node of network. A process may be migrated because the local node does not have the required resources. Over the past few years a number of studies in optimization techniques and their applications to distributed system have led to the identification of several challenging problems. One of these problems is optimally assigning the tasks to the processors in the distributed system. In such system, it is essential to assign each task to the processor whose characteristic are more suitable for the execution of the task. Availability of information at geographically dispersed locations and the need of its transmission from one source to another have also led to the development of distributed system. The task allocation problem in a distributed system may either be static or dynamic. If the allocation is static the information regarding over all costs of a hardware configuration is known before hand. When a task is assign to a particular processor, it stays on that processor during its life time. The problem of assigning m tasks to n processors, where m > n as normally is the case, in the real life is a complex one. The

complexity may be reduced by an efficient task partitioning strategy [BERG 1987]. In case of dynamic allocation the tasks or modules can reassign dynamically during program execution, so as to the advantage of changes in the local reference patterns of the program [MANI 1998]. Although the dynamic allocation has potential performance advantages, static allocation is easier to realize and less complex to operate. Recently, Zhou, Zhibo, Zhou, Tong and Jinxiang Wang [ZHOU 2007] have discussed the bit error rate performance of an Invariant Delay Multiple Access DCSK (IDMA-DCSK) Communication System over multi path fading channels. They have established the close relation between theoretical and the simulation results. Ha, C and Kuo, W. [HA 2006] has also presented the multi-path heuristic for redundancy allocation. Onishi, J., Kimura, S., James, R.J.W., Nakagawa, Y., [ONIS 2007] has solved the redundancy allocation problem by using surrogate constraint method.

2.1.1 Decomposition Approach

The reliability evaluation of complex networks using decomposition method is also one of the important techniques. The reliability of an oriented type-I network is computed by Nakazawa [NAKA 1976] by decomposing the network into short-and-open-removed sub networks according to its keystone line called a removable line. In an another paper [NAKA 1977] the type-I of networks is extended to types-II, II & III and equivalence of one non oriented line and one pair of oriented lines with equal 1-reliability is proved for types I, II & III. It is also proved that the type III networks have to be transformed into a Boolean expression and then the Boolean decomposition method [NAKA 1977] can be applied. A network [NAKA 1978] is represented by matrices which

is suitable for computer use. The criteria for removable lines of complex networks are proved. It is also proved that there are always removable lines of complex networks except oriented type III one, using the matrix expression [NAKA 1981]. For determining the conditional success events Aggarwal et al [AGGA 1982] evolved a technique using both the node removal and connection multiplication methods for path enumeration. Li et al [LI 1992] suggested an algorithm for optimization of large-system reliability with the help of decomposition method.

2.1.2 Genetic Algorithmic Approach

This optimization approach is a heuristic optimization technique, which includes simulated annealing. Tabu searched an evolutionary strategy [REEV 1993]. Coit and Smith [COIT 1996] developed a specific genetic algorithm to analyze series-paralleled system and to determine the optimal design configuration where redundancy allocation problem for a series-parallel system had a specified number of sub systems and for each sub system, there are multiple component choices which can be selected and used in parallel. For those system designed using off- the- shelf component types, with known cost, reliability and weight, the system design and component selection become a combinational optimization problem. They [COIT 1998a] also presented an algorithm to solve the redundancy allocation problem when the objective is to maximize a lower percentile of the system time- to- failure distribution. Levitin et al [LEVI 1998] discussed a redundancy optimization problem to multi-static systems. Here the systems and their components have the range of performance levels. Dengiz et al [DENG 1997] used the similar approach to solve the all-terminal network design problem when considering cost and reliability. Genetic algorithm was also used in optimization of system reliability

[KUMA 1995b, PAIN 1995], while Kumar et al [KUMA 1995a] worked on the topological design for such systems. Exploring Genetic Programming and Boosting Techniques to Model Software Reliability has been studied by Costa et al [COST 2007].

2.1.3   Integer Programming Approach

A task allocation model becomes more significant and realistic when it incorporates real- time constraints such as inter-processor communication, memory limitations of each processor, etc. For the task allocation problems, Integer Programming is a useful and exhaustive approach as it is capable to reflect real- life situations of distributed processing and simplicity in application. A number of researchers have worked on task allocation problems employing this technique to determine the optimum solution [MA 1982, DESS 1980, CHU 1980a]. A model based on this approach is developed by Chu for optimum file allocation in a multiple computer system [CHU 1969]. Another similar approach for data file allocation has been reported by R. Marcogliese and Novarese [MARC 1981].

2.1.4 Dynamic Programming

Dynamic programming is an algorithm design method that can be used when the solution to a problem may be viewed as the result of a sequence of decisions. The main feature of this technique is that it often drastically reduces the amount of enumeration by avoiding the enumeration of some decision sequences that can not possibly be optimal. This approach aims at finding the optimal sequence of decision employing the principal of optimality. Dynamic programming based solution procedure can be regarded as recursive optimization because the decisions, leading to an optimal solution, are made

one at a time. A number of research papers based on dynamic programming approach are available in the literature of task allocation problem in CS. Employing this approach, Bokhari devised a shortest tree algorithm that runs in o(mn^2) time for the assignment of 'm' modules to 'n' processors [BOKH 1981a]. A relationship between module allocation and non – serial dynamic programming has been established by Rosenthal [ROSE 1982]. Considering cost and reliability aspects, Ashrafi et al [ASHR 1992] developed optimization models for selection of programs. Dynamic programming technique is applied by Ferman Dez-Baca [FERN 1989] for obtaining the optimal assignment through local search.

2.1.5   Network Flow and Network Partitioning Algorithm

Network Flow algorithm keeps an important role for the task allocation problems. Stone used this technique for the task allocation in the multiprocessor systems in 1977 [STON 1977] by making use of Ford Fulkersen algorithm [FORD 1962] for finding maximum flows in commodity networks as modified by Edmonds and Karp [EDMO 1972, KARZ 1974, DINI 1970]. The complexity of this algorithm is known to be O [(m+2)^3], where m is the number of tasks. This method provides no mechanism for representing memory size or other constraints. Wu et al [WU 1980a] employed network flow algorithm for task allocation and also described the partition algorithm for parallel and distributed processing [WU 1980b]. In another algorithm load balancing in a circuit-switched multi computer has been used by Bokhari [BOKH 1993]. Ke and Wang [KE 1997] proposed an efficient reliability evaluation algorithm accounting for imperfect nodes in distributed computing networks. Based on the concept of network partition, the algorithm exploits some simple efficient techniques to handle the unreliable nodes, for

directly computing the network reliability expression considering imperfect nodes instead of using any compensating method.

2.1.6 Problem Reduction Method

A smaller problem can be handled more effectively in comparison to a bigger problem. It is therefore, always advisable to reduce the size of a given problem, if possible. A task allocation problem can be simplified to a certain extent by reducing the size of the problem. Arora and Rana, who proposed module assignment in two processor distributed system [AROR 1979], adopted this idea for proposing the concept of clustering the tasks which exhibit certain peculiar behavior to reduce the problem size [AROR 1981]. A task is either assigns to a processor or fused with another task(s) depending upon some criterion. Sagar and Sarje [SAGA 1991] used the above technique for task allocation model for evolving new distributed systems. A clustering algorithm for assignment problem of arbitrary process systems to heterogeneous distributed computer system is implemented by Bowen et al [BOWE 1992]. He showed that clustering algorithm exhibit better time complexity. The clustering technique used by Kim and Browne [KIM 1988] iteratively applies a critical path algorithm to transform the graph into a virtual architecture graph which consists of a set of linear clusters and the interconnection between them. A new heuristic, based on the concept of clustering, to allocate tasks for maximizing reliability is proposed by [SRIN 1999]. Based on clustering approach, a new multiprocessor scheduling technique named de-clustering has been given by Sih and Lee [SIH 1993b]. They claimed that their de-clustering approach not only retains the clustering advantages but at the same time overcomes its drawbacks.

2.1.7 Petri Net Modeling & Reliability Evaluation

The probabilistic graphs, being static in nature can not efficient model and analyze the dynamic behavior of the system [JOLL 1980]. A Petri net approach introduced by Petri [PETR 1962] and Holt [HOLT 1978] is recognized as one of the most powerful modeling and analysis tool for a communicating processing environment. Petri net theory to model a distributed processing environment is also studied by Murata [MURA 1979]. Kumar and Aggarwal [KUMA 1993] suggested Petri net model for the evaluation of reliability for the execution of a computer program in a CS. The execution of a program in a distributed computing system may require access to several files residing at different sites and communications paths between several node pairs. Then, by using the ratability, firing and marking concepts of Petri net, an algorithm is developed to study the distributed computer program reliability and the CS reliability. Earlier Kumar et al [KUMA 1988, KUMA 1990] used this approach to obtain the reliability of a broadcasting network. Lopez [LOPE 1994] presented a model based on stochastic Petri net to estimate the reliability and availability of programs in a DPE. Some models [SHAT 1992, SHAT 1989] for reliability oriented task allocation in redundant distributed computer system and estimating the reliability and availability of programs in a distributed processing environment. Recently Schnee Weiss, W.G. [SCHN 2007] presented a review of Petri net picture book and Petri nets for reliability modeling. The reliability of tree structured grid services has been studies by Yuan – Shun, Dai [YUAN 2006, YUAN 2007]. The study related to confidence bounds for system reliability was presented by Ramirez-Marquiz, J.E. and Wei, Jian [RAMI 2006].

2.1.8 Shortest Path Algorithmic Method

The solution of the task allocation problem for a CS can be obtained by using the shortest path algorithms. Work based on such algorithms is reported in the literature [BOKH 1981a, PRIC 1982, TOWS 1986]. In order to apply this approach, a program graph is transformed into an assignment graph. For each node in the program graph, there exists a set of nodes corresponding to the number of processors in this assignment graph. The nodes and edges between them are labeled suitably depending on the method applied to obtain a shortest path from source- nodes to terminal- node. A shortest tree algorithm described in [BOKH 1981a] minimizes the sum of the execution and communication costs for arbitrarily connected distributed systems with arbitrarily number of processors, provided the interconnection pattern of modules form a tree. The objective of the study was to allocate the tasks, whenever possible, to the processors on which they execute fastest while taking into account the over head due to IPC. Dynamic programming approach was applied for determining the shortest path and then optimal assignment of the tasks of a program over the processors of an inhomogeneous DCS was obtained. The worst- case complexity of the method is found to be o $(n^3 \phi^3)$, where, 'n' is number of nodes in assignment graph and 'ϕ' is the number of phases. Price and Pooch claimed that their shortest path method is applicable to all cases with the limitation that an optimal solution is possible in limited cases only [PRIC 1982]. The shortest path algorithm evaluates the set of nodes in the assignment graph corresponding to a program graph to select the best possible allocation of a task to a processor. Towsley's work [TOWS 1986] is a special cases where the inter task communication pattern is series parallel in nature. A search is made in the assignment graph for the parts of the program graph where inter-

task communication patterns are in series parallel or tree in nature. For such inter task communication patterns, shortest paths are derived and these shortest paths are combined to get the overall shortest path of the assignment graph. In most of the other related studies [BOKH 1987, IQBA 1986a, IQBA 1986b, KOHL 1975, BOKH 1988], and optimal task assignment graph is constructed which contains as many layers as number of processors. The calculation of weights of edges depends on the nature of the program graph and the processor mechanisms. Allocation of task interaction graphs to processor in heterogeneous networks is reported by Hui and Chanson [HUI 1997]. Keinghan [KEIN 1971] used a form of dynamic programming approach by partitioning a chain structured program graph. Bhardwaj et al. [BHAR 1994] have applied a similar method for optimal sequencing and arrangement in distributed single level tree networks with communication delay. Some bottlenecks of the shortest path methods are that the assignment graph becomes very large and complex as the number of processors in the distributed system increases and subsequently number of computations increase for calculating the edge weights. Further to add, the entire assignment graph has to be retained in the memory until the final assignment is made. Ramirez–Marquez, J.E. and Gebre, B.A., [RAMI 2007] suggested a classification tree based approach for the development of minimal cut and path vectors of a capacitated network.

2.1.9. Precedence Constraint

In 2008, Karasakal [KARA 2008] has given a study on air defense missile target allocation model for a naval task group. Task allocation models did not consider modules precedence constraints as one of the significant factor. However, practically, it is not possible to start execution of a task before the completion of another task until the task is

independent. Therefore, besides considering load balancing and minimization of IPC cost the precedence relation among task must also be taken into account. Some studies of task allocation models [CHOU 1986, MARK 1986] considered the precedence relation as a criterion for assignment. The model developed by Markencoff and Liaw [MARK 1986] is applicable for the problems exhibiting parallelism. The job comprising a set of tasks is modeled as a directed a cyclic graph. The graph has been partitioned into a set of sub graphs. Each directed a cyclic graph corresponds to an independent process without any data flow between sub graphs. The distribution of sub directed cyclic graphs, to processors, as evenly as possible, using branch and bound method, results in optimal assignment without IPC cost. The application for the response time of the threads is important rather than the entire application. Chu and Leung [CHU 1984b] reported an algorithm to search for task allocation that minimizes response time. The suggested algorithm considers the task replication and assignment together. Under any given task allocation process, to reduce response time, tasks reallocated from the longest weight processor to shortest weight processors until no further improvement is possible by such task reallocation. An extension for this algorithm with the objective of minimizing response time is proposed by Chu and Lan [CHU 1987b].The objective of task allocation is to counter balance the related but apposite factor of IPC cost and load balancing to achieve maximum throughput and to satisfy real time constraints.

2.1.10. Particular Architecture Consideration

The problem of assigning a set of 'm' tasks to a particular multiprocessor architecture having 'n' processors, where m > n, is known to be a mapping problem. This

problem has been attempted by many researchers [BERM 1984, BERG 1985, BERG 1987, BOKH 1987, FUKU 1987, LEE 1987, MEND 1987, MURT 1988, MURT 1989, SIVA 1988, SIVA 1989, SHMI 1991, CHUA 1992, LIAN 1994, OLSO 1994] to obtain efficient mapping schemes. Chuang et al [CHUA 1992] suggested a fast recognition complete processor allocation strategy for hypercube computers. This model referred to a dynamic processor allocation scheme where the search space generated is dependent upon the dimension of the requested sub cube dynamically, rather than being predetermined and fixed. Olson et al [OLSO 1994] have evolved a model for the fault-tolerant routing in mesh architectures for the distributed computing system. Network flow models of message flow in the mesh and hypercube have been developed by Bokhari [BOKH 1993]. Sih and Lee [SIH 1993a] assumed the target architecture for their proposed scheduling technique to be shared- bus multiprocessor. Another polynomial-time algorithm is developed for computing the distributed program reliability for star topologies of CS in which data files are restricted to a certain type of distribution [LIN 1990].

2.1.11 Reliability Evaluation Consideration

Distributed systems are increasingly being employed for critical applications, such as aircraft control, industrial process control, and banking systems. Maximizing performance has been the conventional objective in the allocation of tasks for such systems. There are many measures to evaluate the performances of distributed system. Reliability of a distributed system is another vary important issue that need special attention [YANG 2008, AGGA 1982, GARC 1982, SEGE 1994]. Yang et al [YANG

2008] have presented a study of cost oriented task allocation and hardware redundancy policies in heterogeneous distributed computing systems considering software reliability. A reliability measure called, distributed program reliability, to model accurately the reliability of CS has also been reported [KUMA 1988]. Some models of reliability oriented task allocation in redundant distributed computer systems [SHAT 1989. SHAT 1992] and estimating the reliability and availability of programs have also appeared in the literature. A 1- step algorithm GEAR (Generalized Evaluation Algorithm for Reliability), capable of computing several reliability parameters of a CS such a terminal-pair reliability, computer network reliability, distributed program reliability, and distributed system reliability, is also studied [KUMA 1993]. Software reliability allocation, based on several factors such as structure, utility, price, and cost, is proposed by Zahedi [ZAHE 1991]. Strategies are devised that achieve a maximal combination of safety and reliability [RAI 1982]. Efficient methods to optimize the system reliability have been reported in [PAIN 1995, RAGH 1985]. Reliability analysis for a distributed program is mentioned in [KUMA 1996, KE 1997]. A genetic algorithm based reliability optimization model for computer networks is proposed by Kumar et al [KUMA 1995a]. Further, an investigation of the problem of distributed- program reliability in various classes of distributed computing systems has been presented [LIN 1990]. Srinivasan and Jha suggested a new clustering-based safety and reliability driven   heuristic for a heterogeneous distributed system [SRIN 1999].

2.1.12 Heuristic Approach

Heuristic Algorithm has also played a vital role for providing solutions to different types of problems. Several research papers [DENG 1997, LU 1986, WARD 1984, PRIC 1984, WILL 1983, ALFO 1988, HWAN 1993, EFE 1982, AROR 1980] have been reported in the literature regarding heuristic approaches for task allocation problem of these systems. The use of CS has been rapidly increasing in order to share expensive hardware and software resource and provides access to main systems from the distant locations. Dengiz et al [DENG 1997] suggested a heuristic algorithm inspired by evolutionary approach to solve the all terminal network design problem when considering cost and reliability. Kartik and Murthy [KART 1995] provided a heuristic algorithm to find an optimal and sub optimal task allocation in redundant distributed computing systems, to maximizing system reliability. Further it repeatedly consider all the possible reassignments of tasks at a time, performs the most advantages reassignments and terminals when no more profile reassignments is discovered. On the other hand, the problem of minimizing the cost by making the clusters of those tasks that is heavily communicating and assigning the tasks cluster to appropriate processors. On the basis of execution drew considerable attention [WILL 1983, PRIC 1984, WARD 1984]. Some heuristic approaches for task scheduling based on characterization of inter module communications [LAN 1985] and reliability maximization [HWAN 1993] in the distributed computing system also deserves mention. The heuristic load balancing approaches [BOLC 1983, YANG 1987] suggested reducing the number of tasks by forming task clusters equal to the number of processors. This reduction is accomplished by fusing those tasks between them data exchanges are maximum. The task-clusters so

formed are assigned to the processors. Finally tasks are shifted from the heavily loaded processors to the minimally loaded processors to balance the load on all the processors. Liu X. et al [LIU 1999] has presented a study of building systems from components to answer the questions related the configuration of components and performance relation with monolithic system. They has also discussed the generality of the technique with regard to systems other than CS. Wang, N, Lu, J.C. and Kvam, P [WANG 2007] has also given the reliability modeling in spatially distributed logistics systems. Another simple heuristic algorithm for generating all minimal paths has been suggested by Yen Wei Chang [YEN 2007]. Recently, Kumar et al [KUMA 2009] have discussed a maximizing business value by optimal assignment of jobs to resources in grid computing.

RESEARCH MODELS

3.1 MOTIVATION

Over the past two and half decades, advancements in microelectronic technology have resulted in the availability of fast, inexpensive processors and advancements in communication technology have resulted in the availability of cost effective and highly efficient computer networks. The net result of the advancements in these two technologies is that the price performance ratio has now changed to favor the use of interconnected multiple processors in place of a single, high speed processor. There are many ways to define distributed system, and also many different levels, types of distributed system models and distributed application development techniques. Various vendors have created and marketed distributed computing systems for years, and numerous initiatives and architectures have been developed to permit distributed processing of data and objects across a network of connected systems. A system in which a large number of separate but interconnected computers do the jobs is called distributed network. In distributed network, services reside at multiple sites. Instead of single large processor being responsible for all aspects of process, there are several separate processor handles these aspects. A distributed network is looks like a virtual uniprocessor. In the distributed networking the program or tasks are also often developed with the subsets of independent units under distributed environments. The performance of distributed network depends mainly on

the number of users, type of transmission medium, hardware, software, etc; while the reliability depends upon frequency of failure, recovery time of network after a failure, etc.

Distributed system utilizes a network of many computers, each accomplishing a portion of a task, to achieve a computational result much more quickly than with a single computer. In addition to a higher level of computing power, distributed computing also allows many users to interact and connect openly. Different forms of distributed computing allow for different levels of openness, with most people accepting that a higher degree of openness in a distributed system is beneficial. The segment of the internet the people are most familiar with, the World Wide Web, is also the most recognizable use of distributed computing in the public arena. Many different computers make everything one does while browsing the internet possible, with each computer assigned a special role within the system. The best-known research problem for such systems is the task assignment problem, in which the total system time is to be minimized. The problem finding an assignment of tasks to nodes that minimizes total execution and communication cost (or time) was elegantly analyzed using a network flow model and network flow algorithms by [STON 1977, STON 1978] and a number of other researchers [KUMA 2007,YADA 2008,YADA 2002, YADA 2003, KUMA 2001, SING 1999, SRIN 1999, PENG 1997, KUMA 1995a, KUMA 1995b, KUMA 1995c, KUMA 1995d, SAGA 1991, ZAHE 1991, BACA 1989, CHU 1969, KUMA 1999, KUMA 1996, KUMA 2002, KUMA 2006, CASA 1988, BOKH 1979] through various different algorithms. The load-balancing algorithms or load-leveling algorithms are based on the intuition that, for better

resource utilization, it is desirable for the load in a distributed system to be balanced evenly. Thus a load-balancing algorithm tries to balance the total system load by transparently transferring the work load from heavily loaded nodes to lightly loaded nodes in an attempt to ensure good overall performance relative to some specific metric of system performance. When considering performance from the user point of view, the metric involved is often the response time of the processors. However, when performance is considered from the resource point of view, the metric involved is the total system throughput [SINH 2004]. Kumar et al [KUMA 1995a] suggested a modified and efficient task allocation approach in which authors have optimized the overall system cost that include the execution and communication cost. Kumar et al [KUMA 1996] and Kumar [KUMA 1999] have suggested task allocation technique, which are dynamic in nature for optimizing the cost and reliability respectively. S. Srinivasan [SRIN 1999] stated that software or a program to be run on the distributed computing system is called a task, which is composed of intercommunication tasks, which allocated to the different processor in the system. Several other methods have been reported in the literature, such as, Integer Programming [DESS 1980], Branch and Bound technique [RICH 1982], Matrix Reduction technique [SAGA 1991], Reliability Optimization [LIN 2002, LYU 2002] Modeling [BIER 2002, FITZ 2002]. The Series Parallel Redundancy-Allocation problem has been studied with different approaches, such as, Non-Linear techniques [TILL 1977], and Heuristic techniques [COIT 1996, COIT 1998a, COIT 1998b, LO 1988].

Therefore it is recorded that the research problem for such systems is of the type tasks allocation problems, in which either system reliability is to be maximized or total system cost is to be minimized or overall effectiveness of the distributed

systems is to be optimized under the pre specified constrains. Sager et al [SAGA 1991] discussed the tasks allocation problem in 1991; they considered a set of tasks and a set of processors, which are less in comparison to the number of tasks. They have suggested a heuristics approach by using matrix reduction technique for the allocation of tasks to the various processors, and obtained optimal execution cost. Kumar et al [KUMA 1995c, KUMA 1996] has studied the task allocation problem for a set of heterogeneous processors on which the set of tasks to be allocated in such a way that all the tasks got executed and load on each processor also maintained. These models gave the total assignment cost and results were much better as compared to Sagar et al [SAGA 1991]. The complexity of the algorithm is also less as they have mentioned the complexity comparisons to other approaches in their papers. Kumar [KUMA 2001] and Singh et al [SING 1999] also extended the work on tasks allocation problem for analyzing the various performance parameters. Recently, Yuan-Shun and Levitin [YUAN 2007] has used optimal resource allocation for maximizing performance and reliability in tree structured grid series. The genetic algorithm is used for reliability-oriented task assignment by Chin-Ching Chiu et al [CHIN 2006]. The work related to redundancy allocation problem with a mix of components is studied by Onishi et al [ONIS 2007].

3.2. ASSUMPTIONS

To cope up with the real life problems and to keep the algorithms reasonable in size the following assumptions are made:

➤ The number of tasks to be allocated is more than the number of processors, as normally is the case, in the real life distributed system.

➤ Whenever two or more tasks are assigned to the same processor the Inter Tasks Communication cost between them is assumed to be zero.

➤ Whenever two or more tasks are assigned to the same processor the Inter Tasks Communication reliability between them is assumed to be one.

➤ Whenever two or more tasks are assigned to the Inter Processor Distance between them is assumed to be zero.

➤ If a task is not executable on a certain processor due to absence of some resources, the processing cost of the task is taken to be infinite (∞).

➤ If a task is not executable on a certain processor due to absence of some resources, the processing reliability of the task is taken to be zero (0).

MODEL-I

RELIABILITY BASED ALGORITHM FOR PERFORMANCE ENHANCEMENT OF DISTRIBUTED SYSTEMS

3.1.1 Objective

The objective of this study is to maximize the overall processing reliability for a distributed system through optimally assigning the tasks of various processor of the distributed system, So that the performance of the distributed system is to be enhanced. The type of assignment of tasks to the processor is static. It also takes care processing of all the tasks, as tasks are more than the numbers of processors of the system. Let the given distributed system consists of a set $P = \{p_1, p_2, p_3 \, p_4, ..., p_n\}$ of n processors, interconnected by communication links and a set $T = \{t_1, t_2, t_3, t_4, ..., t_m\}$ of m tasks. The processing reliability of individual tasks corresponding to each processor are given in the form of matrix PRM (,) of order m x n and the communication is taken in the square symmetric matrices CM (,) of order n respectively. A procedure to assign all the tasks to the processors of the communicating systems based on processing reliability is to be designed in such a way that the overall reliability is to be optimizing under the pre specified constraints. The load on each processor has been also taken care off in order to balance the load of each processor.

3.1.2 Technique

In order to evaluate the overall optimal processing reliability of a distributed system, we have chosen the problem where a set $P = \{p_1, p_2, p_3, ...p_n\}$ of 'n' processors and a set $T = \{t_1, t_2, t_3, ...t_m\}$ of 'm' tasks, where m>n. The processing

reliability of each task to each and every processor is known and it is mentioned in the Processing Reliability Matrix namely PRM (,) of order m x n. Communication amongst the task is also taken into consideration, it is always in the form of 0 (zero) or 1 (one), showing no communication for the case of 0 (zero), while there is a communication for the case of 1 (one). It has been given in communication matrix namely CM (,) of order m x m.

For the each task, we shall be forming the cluster, it will contain those set of task that have communication with this task. The number of tasks in a cluster is [m/n] for the case m is even, otherwise [(m+1)/n]. The tasks those are not having any communication with another tasks are called isolated tasks. Rearranging the set of clusters, in such a way that number of ordered pair should be n and none of the task is to be repeated in any ordered pair of the cluster. Each and every ordered pair of the cluster shall be containing at least one task; remaining ordered pair may be neglected.

Now, modify the Processing Reliability Matrix PRM (,) by multiplying row(s) according to the task(s) as appear in the ordered pair of the cluster. It will give us a modified Processor Reliability Matrix PRM (,) of order n x n. In order to get the optimal assignment, we have used the modified version of row and column assignment method proposed by Kumar et al [KUMA 95c], in which first of all we subtract all of the elements of PRM (,) by 1, by which we obtain unreliability matrix. Finally the function to obtain the overall processing reliability (*Preliability*) of the unreliability matrix of the distributed system is defined as mentioned given below;

$$\mathrm{Pr}\, elibility = \left[\prod_{i=1}^{n} \left\{ \sum_{j=1}^{n} ER_{ij} x_{ij} \right\} \right] \qquad (1)$$

Where, $x_{ij} = \begin{cases} 1, \text{if } \textit{ith task is assigned} \text{ to } \textit{jth} \text{ processor} \\ 0, \text{otherwise} \end{cases}$

3.1.3 Algorithm

To given an algorithmic representation to the technique mentioned in the previous section, let us consider a system in which a set of m tasks T= $\{t_1, t_2, t_3, \ldots, t_m\}$ is to be executed on a set of n available processors P= $\{p_1, p_2, p_3, \ldots, p_n\}$.

Step 1:   Start algo

Step 2:   Read the number of tasks in m

Step 3:   Read the number of processors in n

Step 4:   For I = 1 to n

Step 5:   For J = 1 to m

Step 6:   Read the value in PRM [I][J]

Step 7:   Increase the value of J by 1

Step 8:   End of J loop

Step 9:   Increase the value of I by 1

Step 10:   End of I loop

Step 11:   For I = 1 to n

Step 12:   For J = 1 to n

Step 13:   Read the value in CM [I][J]

Step 14:   Increase the value of J by 1

Step 15:   End of J loop

Step 16:   Increase the value of I by 1

Step 17:   End of I loop

Step 18:   For I = 1 to n

Step 19:   For J = I to n

Step 20:   If CM[I][J] == 1 then

Step 21:   Store the value of P1[I] to 1

Step 22:   Store the value of P2[J] to 1

Step 23:   Calculate MAT [I][J] = PRM[P1[I]] * PRM [P2[J]]

Step 24:   End of if statement

Step 25:   Increment the value of J by 1

Step 26:   End of J loop

Step 27:   Increase the value of I by 1

Step 28:   End of I loop

Step 29:   For I = 1 to n

Step 30:   If P1[I] ==0 then

Step 31:   Store the value of T1[I] by P1[I]

Step 32:   End of if statement

Step 33:   Increase the value of I by 1

Step 34:   End of I loop

Step 35:   For I = 1 to n

Step 36:   If T1[I] !=0 then

Step 37:   Calculate MAT [T1[I]] = PRM [T1[I]] * PRM[T1[I+1]]

Step 38:   End of if statement

Step 39:   Increment the value I by 1

Step 40:   End of I loop

Step 41:   For I = 1 to n

Step 42:   For J = 1 to n

Step 43:   MAT [I][J] = 1 – MAT [I][J]

Step 44:   Increase then value of J by 1

Step 45:   End of J loop

Step 46:   Increase the value of I by 1

Step 47:   End of I loop

Step 48:   MIN = MAT [I][J]

Step 49:   For I = 1 to n

Step 50:   For J = 1 to n

Step 51:   If MAT [I][J] < MIN then

Step 52:   MIN = MAT [I][J]

Step 53:   Endif

Step 54:   Increase the value of J by 1

Step 55:   End of J loop

Step 56:   MIN = 0

Step 57:   Increase the value of I by 1

Step 58:   End of I loop

Step 59:   Count the zero(s) in each row

Step 60:   Mark the row(s), which have single zero

Step 61:   Mark the column, which have single zero

Step 62:   Go to the row(s), which have more than one zero. Now select any one zero

              and cross the leading zero(s), which are in same row and column

Step 63:   Mark the assignments

Step 64:   Count   the total assignment

Step 65:   If total number of assignment < order of matrix

Step 66:   Go to Step 70

Step 67:   Else

Step 68:   Go to Step 77

Step 69:   End of if statement

Step 70:   Mark the rows for which assignment have not been made

Step 71:   Mark column that have zeros in marked rows

Step 72:   Mark rows that have assignment in marked column

Step 73:   Repeat Step 71 & Step 72 until chain of marking ends

Step 74:   Draw the minimum number of lines through unmarked rows and marked columns to cover all zeros

Step 75:   Select the smallest element of the uncovered elements and replace it by zero. Also add this element to positions at which lines intersect to each other only

Step 76:   Go to Step 60

Step 77:   State processing reliability

Step 78:   End algo.

3.1.4 Implementation

Example-I

In the present model-I, the distributed system consist a set P of 3 processors $\{p_1, p_2, p_3\}$ and a set T of 5 tasks $\{t_1, t_2, t_3, t_4, t_5\}$. It is shown by figure 1. The processing reliability of each task on various processors are known and mentioned in the matrix namely PRM (,).

Figure 1
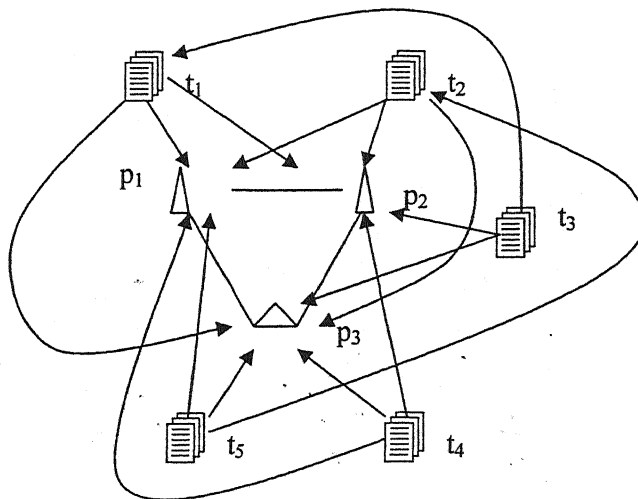
$$
PRM(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.999856 & 0.999420 & 0.999635 \\ 0.999632 & 0.999632 & 0.999120 \\ 0.999235 & 0.999541 & 0.999631 \\ 0.999632 & 0.999520 & 0.999687 \\ 0.999863 & 0.999235 & 0.999631 \end{array}
$$

The communication amongst the tasks has also taken into consideration and it is shown by figure 2. Its matrix representation has been given by the communication matrix namely CM(,).

Figure 2

$$
CM(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \\ 0 & 1 & 1 & 1 & 0 \\ & 0 & 0 & 1 & 0 \\ & & 0 & 1 & 1 \\ & & & 0 & 0 \\ & & & & 0 \end{array}
$$
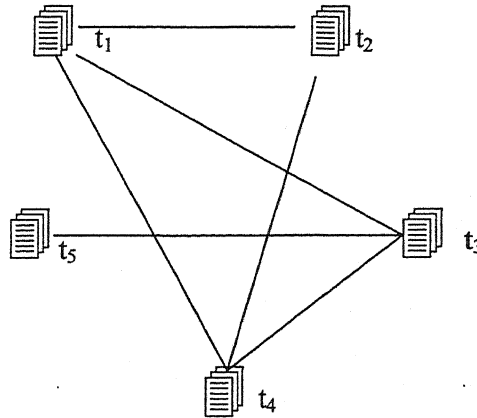
On the basis of the above communication matrix CM (,) it is clear that following clusters can be formed depending upon the existence of the communication.

For task t_1 : {(t_1 * t_2), (t_1 * t_3), (t_1 * t_4)}
For task t_2 : {(t_2 * t_4)}
For task t_3 : {(t_3 * t_4), (t_3 * t_5)}

For task t_4 : −

For task t_5 : −

From the above sets of clusters by neglecting repeated tasks or ordered pair we get (t_1 * t_2) and (t_3 * t_4). On clubbing the values corresponding these set of tasks or ordered pair we get Final Processor Task Matrix namely FPTM (,):

$$FPTM(,) = \begin{array}{c} \\ t_1*t_2 \\ t_3*t_4 \\ t_5 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ \left[\begin{array}{ccc} 0.999488 & 0.999052 & 0.998755 \\ 0.998867 & 0.999061 & 0.999318 \\ 0.999863 & 0.999235 & 0.999631 \end{array}\right] \end{array}$$

Now on applying Kumar et al [KUMA 95c] strategy, we obtain unreliability matrix after that we obtain optimal assignments and optimal effective reliability as mentioned below.

Processor	Task assigned
p_1	$t_1 * t_2$
p_2	$t_3 * t_4$
p_3	t_5

Optimal effective reliability = 0.996860. The graphical representation of the optimal assignment is shown by the figure 3.
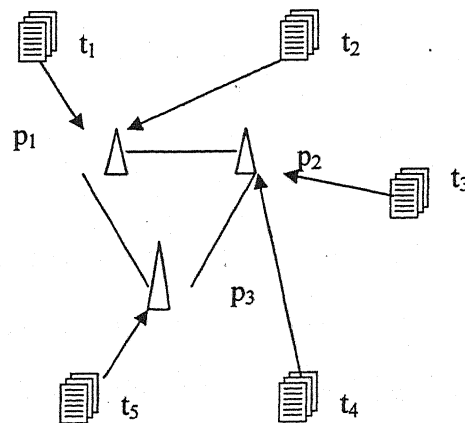
Figure 3

3.1.5 Conclusion

In this research model we have chosen the problem, in which the number of the tasks is more than the number of processors of the distributed system. The model mentioned here is based on the consideration of processing reliability of the tasks to various processors. The method is presented in pseudo code and implemented on the several sets of input data to test the performance and effectiveness of the pseudo code. It is the common requirement for any assignment problem that the tasks have to be processed with maximum reliability. Here, performance is measured in terms of processing reliability of the task that has been processed by the processors of the network and also these tasks have been processed optimally. The optimal result of the example that is considered to test the algorithm and it is mentioned in the implementation section of the paper are as given below.

Processor	Task	Optimal result
p_1	$t_1 * t_2$	
p_2	$t_3 * t_4$	0.996860
p_3	t_5	

As we know that, the analysis of an algorithm is mainly focuses on time complexity. Time complexity is a function of input size 'n'. It is referred to as the amount of time required by an algorithm to run to completion. The time complexity of the above mentioned algorithm is $O(m^2n^2)$. By taking several input examples, the above algorithm returns following results,

No. of processors (n)	No. of tasks (m)	Optimal Results
3	4	144
3	5	225
3	6	324
3	7	441
3	8	576
4	5	400
4	6	576
4	7	784
4	8	1024
4	9	1296
5	6	900
5	7	1225
5	8	1600
5	9	2025
5	10	2500

The graphical representation of the above results are shown by figure 4, 5 and 6 as mentioned below,
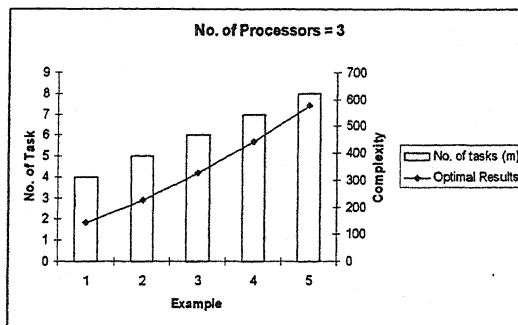
Figure 4
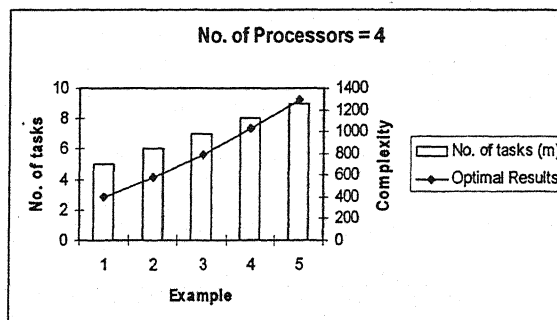
Figure 5
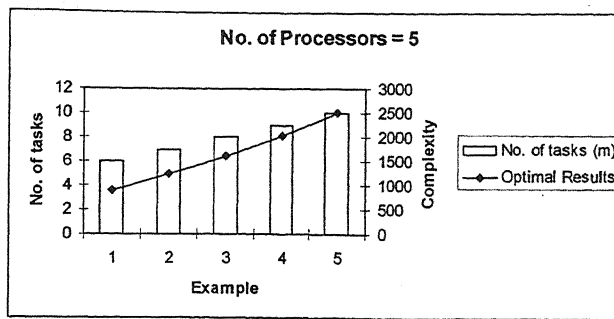
Figure 6

The performance of the algorithm is compared with the algorithm suggested by Richard et al [RICH 1982]. Following table shows the time complexity comparison between algorithm (Rich) and present algorithm.

Processors n	Tasks m	Time Complexity of algorithm [RICH 1982] $O(n^m)$	Time Complexity of present algorithm $O(m^2n^2)$
3	4	81	144
3	5	243	225
3	6	729	324
3	7	2187	441
3	8	6561	576
4	5	1024	400
4	6	4096	576
4	7	16384	784
4	8	65536	1024
4	9	262144	1296
5	6	15625	900
5	7	78125	1225
5	8	390625	1600
5	9	1953125	2025
5	10	9765625	2500

From the above table it is clear that present algorithm is much better for optimal allocation of tasks that upgrade the performance of distributed network. Following graphs shows the, pictorial representation between algorithm [RICH 1982] and present algorithm.
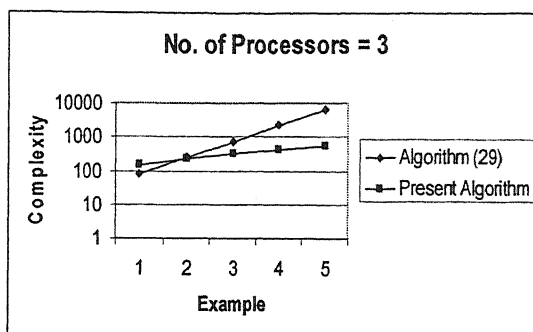
Figure 7
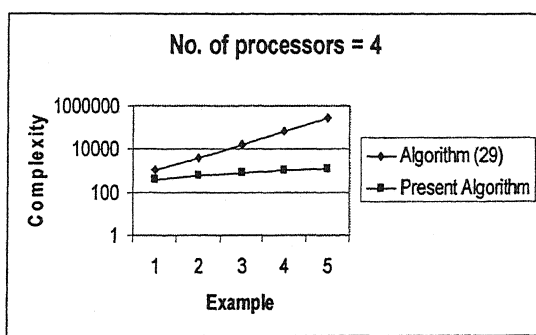
Figure 8
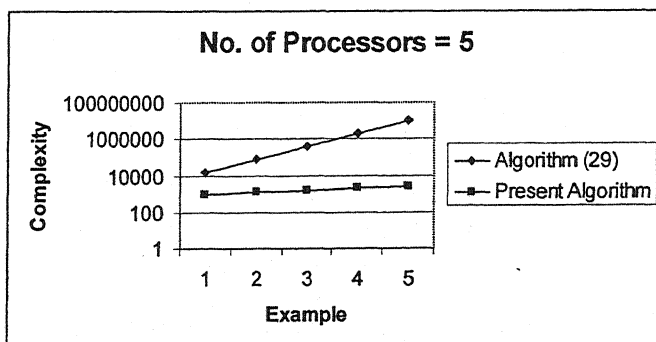
Figure 9

Model-II

TASKS ALLOCATION THROUGH RELIABILITY

INDEX FOR DISTRIBUTED SYSTEMS

3.2.1 Objective

A set of "m" tasks $T=\{t_1,t_2,....,t_m\}$ and a set of "n" processor $P=\{p_1,p_2,p_3,....p_n\}$, interconnected by communications links are assumed in a distributed system. The inter task communication cost of the tasks have been taken in to consideration, which is mentioned in the square matrix namely ITCCM(,) of the order m. Other two factors reliability and cost of each task to each processors are defined in the matrices PRM(,) and PCM(,) of the order m x n respectively. In order to achieve the goal of algorithm, following steps have been considered:

- Fusion criteria for the remaining "(m-n)" task(s) have been carried out.

- Total reliability and total cost of the allocations has been calculated.

- Optimal cost-based reliability index has been defined, which indicates the final (optimal) allocations.

- The criteria for Load balancing are taken into consideration.

3.2.2 Technique

To assign tasks to processors in a distributed system, we have considered the inter task communication cost, Processing cost and Processing Reliability in the form of a square matrix ITCCM (,) of order m, PCM(,) and PRM(,) matrices each of order m x n where, (m>n) respectively, where m represent the number of tasks and n is the number of processors. In the approach suggested here, "m-n" task (s), are to be fused with some other task (s), where the ITCC of $(m-n)^{th}$ task (s) is maximum with some

other task (s). The process shall be continue until all the "m-n" task (s) get fused. Corresponding to the , fused task (s) row (s) we modify PCM (,) PRM(,) and ITCCM (,) matrices. To modify reliability matrix, we multiply the concerning row (S) and deleting that row (s) which corresponds to the fused task (s), while in the processing cost matrix, we add the respective row (s) and deleting the fused task (s) row (s). This processes reduce PRM (,) and PCM(,) into square matrices of order n. Now in order to modify ITCCM(,), we replace all the entries of fused task(s) row (s) by zero, and adding the row(s) to the concerning row(s). ITCCM(,) is also reduced into a square matrix of order n. The assignments can be made by generating all (n!) possible combinations of processors, corresponding to task(s) or a set of task(s). Allocation of task(s) to processor will be made in sequence to avoid the unbalanced utilization of processor, finally, the total processing reliability, and total processing cost of each combination is to be calculated by using the following relations:

$$
\text{Total Processing Reliability (TRP)} = \prod_{i=1}^{m} \prod_{j=1}^{n} x_{ij}\, R_{ij}
$$

and Total Processing Cost (TPC) =

$$
\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}\, c_{ij} \;+\; \sum_{i=1}^{m} \sum_{j=1}^{m} x_{ij}\, cc_{ij}
$$

and,

$$
RI = \text{Reliability functions x inverse of cost function}
$$

$$= TPR \times 1/TPC$$

$$\left[\prod_{i=1}^{m} \prod_{j=1}^{n} x_{ij} \, r_{ij} \right] \quad x \quad 1/\left[\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} \, c_{ij} + \sum_{i=1}^{m} \sum_{j=1}^{m} x_{ij} \, cc_{ij} \right]$$

respectively.

The maximum value of reliability index represents the optimum reliability and optimal cost as well as optimal assignment.

3.2.3 Computational Algorithm

To given an algorithmic representation to the proposed method as discussed, we have to concentrate on such a system which consist a set $P = \{p_1, p_2, \ldots, p_n\}$ of n processor and a set $T = \{t_1, t_2, \ldots, t_m\}$ of m executable tasks which are to be processed by any one of the processor of the distributed system.

Step-1: Input : m,n, PRM(,), PCM(,), ITCCM(,)
      nar := 0;
      RLM :=1.0;
      COA := 0.0;

Step-2: IPROD :=n!
      nlp :- m-n
      These "m-n" task(s) are fused with some other task(s), where inter task communication cost is maximum, and this process is continued till all the task(s) get fused.

Step-3: for i : =1 to n do
      for j : = i +1 to n do
      begin

Step 3.1:   Arrange the upper diagonal values of ITCCM(,) in descending order and store the result in MCC(),an array of order MN =(m-1)*(n-1).

Step 3.2: Store the corresponding position in POS [i , j], where, i = 1........MN

65

and j=1,2.
    end;

Step-4: for i:=1, nlp do
        for j:= 1,2 do
        begin

Step 4.1: Pick-up the task corresponding to maximum position and store the
          task(s) combinations along with the fused tasks in the array TCOMB(i).
          end;

Step-4.2: for i: =1 to m do
          for j: =1 to n do
          begin
          Modify PRM(,) by multiplying the ith and k^{th} row, and delete k^{th} row. Store the
          modified PRM(,) in NPRM(,) of order n. Also, modify the PCM(,) by adding
          the ith row to k^{th} row, and deleting k^{th} row. And thus store the modified PCM(,)
          in NPCM(,) of order n.

Step-4.3: for i: =1 to m do
           for j: =1 to n do
          Replace row and column entities corresponding to {i, k} by zero in ITCCM(,)
          and then add i^{th} row to k^{th} row and i^{th} column to k^{th} column and deleting k^{th} row
          and k^{th} column from the ITCCM(,). It modified the ITCCM(,), store the result
          in NITCCM(,).
          end;

Step-5: Evaluate communication cost, as the sum of upper diagonal element of
        NITCCM(,).

Step-6: for i:=1 to IPROD do
        begin

Step-6.1: Generate the combination of processors and store these in the array
          PCOMB(i).

Step-6.2: Generate a combination to assign the task(s) from TCOMB(j) to processor
          PCOMB(i).

Step-6.3: nar: =nar +1,
          Store the assignment combinations in COMBP(nar,n).

Step-6.4: Obtain RLM by multiplying the task execution reliabilities corresponding to
          the allocated processor(s) as given in PRM(,) and store this value in the array
          TRL(nar).

Step-6.5: Find the value of COA by adding the corresponding execution costs of tasks
          at the allocated processor(s) as shown in PCM(,) and communication cost.
          Store this cost in the array TCO(nar).

Step 6.6: RI(nar):= [TRL (nar)x{1/TCO(nar)}]

Step-7: If nar<IPROD
     Goto step-6.1.
     end;
Step-8: for i:= 1 to nar do

     begin
     MRI :=max[RI (i)];
     Write, OCOMB : =COMBP(i,n);
     ORL :=TRL(i);
     OCO : = TCO(i).
     end;

Step-9: Stop.

3.2.4 Example

Let us consider the distributed system, consisting a set $T = \{t_1, t_2, t_3, t_4\}$ of

4 tasks and a set $P=\{p_1, p_2, p_3\}$ of 3 processors.

Step-1 :

$$
PRM(,) := \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array}
\begin{array}{ccc}
p_1 & p_2 & p_3 \\
\left[\begin{array}{ccc}
0.98 & 0.99 & 0.97 \\
0.96 & 0.98 & 0.99 \\
0.99 & 0.97 & 0.98 \\
0.97 & 0.96 & 0.99
\end{array}\right]
\end{array}
$$

$$
\text{PCM}(,) :=
\begin{array}{c}
\\ t_1 \\ t_2 \\ t_3 \\ t_4
\end{array}
\begin{array}{ccc}
p_1 & p_2 & p_3 \\
\left[\begin{array}{ccc}
9 & 10 & 8 \\
7 & 9 & 10 \\
10 & 8 & 9 \\
8 & 7 & 10
\end{array}\right]
\end{array}
$$

$$
\text{ITCCM}(,) :=
\begin{array}{c}
\\ t_1 \\ t_2 \\ t_3 \\ t_4
\end{array}
\begin{array}{cccc}
t_1 & t_2 & t_3 & t_4 \\
\left[\begin{array}{cccc}
0 & 2 & 5 & 8 \\
2 & 0 & 3 & 4 \\
5 & 3 & 0 & 6 \\
8 & 4 & 6 & 0
\end{array}\right]
\end{array}
$$

Step-2: IPROD := 3! = 6

   nlp   := 4-3   = 1

Step-3: for i:= 1 to 4 do

   for j;= 2 to 4 do

Step 3.1: MCC( ) = [8,6,5,4,3,2]

Step-3.2:

POS[i,j] :-   [(1,4), (3,4), (1,3), (2,4),(2,3),(1,2)]

Step-4: for i := 1 to 1 do

   for j := 1 to 2 do

68

Step 4.1 : TCOMB(1) : = [t_1*t_4 t_2 t_3]

repeat

repeat

Step-4.2 : for i := 1 to 4 do

    for j := 1 to 3 do

Tasks t_1 and t_4 are fused. So we have to modify PRM(,) by multiplying the 1st row to 4th row and deleting the 4th row from PRM(,). Store these results in NPRM(,), we have,

$$
\text{NPCM(,) : =} \quad
\begin{array}{c}
\\
t_1*t_4 \\
\\
t_4 \\
\\
t_3
\end{array}
\begin{array}{ccc}
p_1 & p_2 & p_3 \\
\left[\begin{array}{ccc}
0.9506 & 0.9504 & 0.9603 \\
0.96 & 0.98 & 0.99 \\
0.99 & 0.97 & 0.98
\end{array} \right]
\end{array}
$$

Step 4.3 : for i : = 1 to 4 do

    for j : = 1 to 3 do

Tasks t_1 and t_4 are to be fused, therefore modify PCM(,) by adding 1st row to the 4th row and deleting the 4th row from PCM(,), store the result in NPCM(,), we have,

$$
\text{NPCM(,) : =} \quad
\begin{array}{c}
\\
t_1*t_4 \\
\\
t_2 \\
\\
t_3
\end{array}
\begin{array}{ccc}
p_1 & p_2 & p_3 \\
\left[\begin{array}{ccc}
17 & 17 & 18 \\
7 & 9 & 10 \\
10 & 8 & 9
\end{array} \right]
\end{array}
$$

Step-4.4 :

for i : = 1 to 4 do

for j : = 1 to 4 do

Tasks t_1 and t_4 are to be fused. Therefore replace row and column entries corresponding to 4th row and column by zero in ITCCM(,) and then add 1st row to 4th row and 1st column to 4th column and also deleting 4th row and 4th column from ITCCM(,) & then modified the ITCCM(,), store the result in NITCCM(,). We have,

$$
\text{NITCCM(,)} := \begin{array}{c} \\ t_1{*}t_4 \\ \\ t_2 \\ \\ t_3 \end{array} \begin{array}{ccc} t_1{*}t_4 & t_2 & t_3 \\ \left[\begin{array}{ccc} 0 & 2 & 5 \\ 2 & 0 & 3 \\ 5 & 3 & 0 \end{array}\right] \end{array}
$$

Step-5: Evaluate communication cost as the sum of upper diagonal

elements of NITCCM(,) i.e.CC : = 10

Step- 6 : for i := 1 to 6 do

Step- 6.1 to 6.6 :

nar : = 1

$$t_1{*}t_4 \quad t_2 \quad t_3$$

COMBP(1,3) : = [p_1 p_2 p_3]

TRL(1)   : =   9.129562E-1

TCO (1)   : =   45

RI(1)   : =   2.02879E-2

Step –7: On repeating steps 6.1 to 6.6 we get the following processors combinations as well as cost, reliability and reliability index for various task combinations:

Processor Combination	$t_1 * t_4$ t_2 t_3	Total Processing Cost	Total Processing Reliability	Reliability Index
COMBP (1,3)	1 2 3	45	9.129562E-1	2.02879E-2
COMBP (2,3)	1 3 2	45	9.128611E-1	2.02858E-2
COMBP (3,3)	2 1 3	43	8.941363E-1	2.07938E-2
COMBP (4,3)	2 3 1	47	9.314870E-1	1.98188E-2
COMBP (5,3)	3 1 2	43	8.942313E-1	2.07960E-2
COMBP (6,3)	3 2 1	47	9.316830E-1	1.98230E-2

Step – 8:   The final assignments corresponding to the maximum value of reliability index (optimal index) along with the reliability and cost are shown in the following table.

Optimal Allocation	Optimal Reliability	Optimal Cost	Optimal Reliability Index
$t_1 * t_4 \rightarrow p_3$ $t_2 \rightarrow p_1$ $t_3 \rightarrow p_2$	8.942313E-1	43	2.07960E-2

Step – 9 : Stop

3.2.5 Conclusion

In this research model an algorithm for optimal tasks allocation based on reliability index, define for the purpose, in a distributed system is suggested. The present algorithm determines the total processing reliability and total processing cost corresponding to optimal assignment. The influence of the communication cost further reduces the combination as compared to Kumar, et al [Kuma 99]. The reliability and cost corresponding to maximum reliability index gives the optimal reliability and optimal cost of the optimal assignment. The various stages of allocation have shown through figures 1 to 3. A comparison amongst the complexities of model presented in this chapter with earlier models given by Zahedi [ZAHE 1991] and Kumar et. al [KUMA 99] has been considered. The model may be appreciated at the time complexity front which is of order O (n!) as is evident from the table:

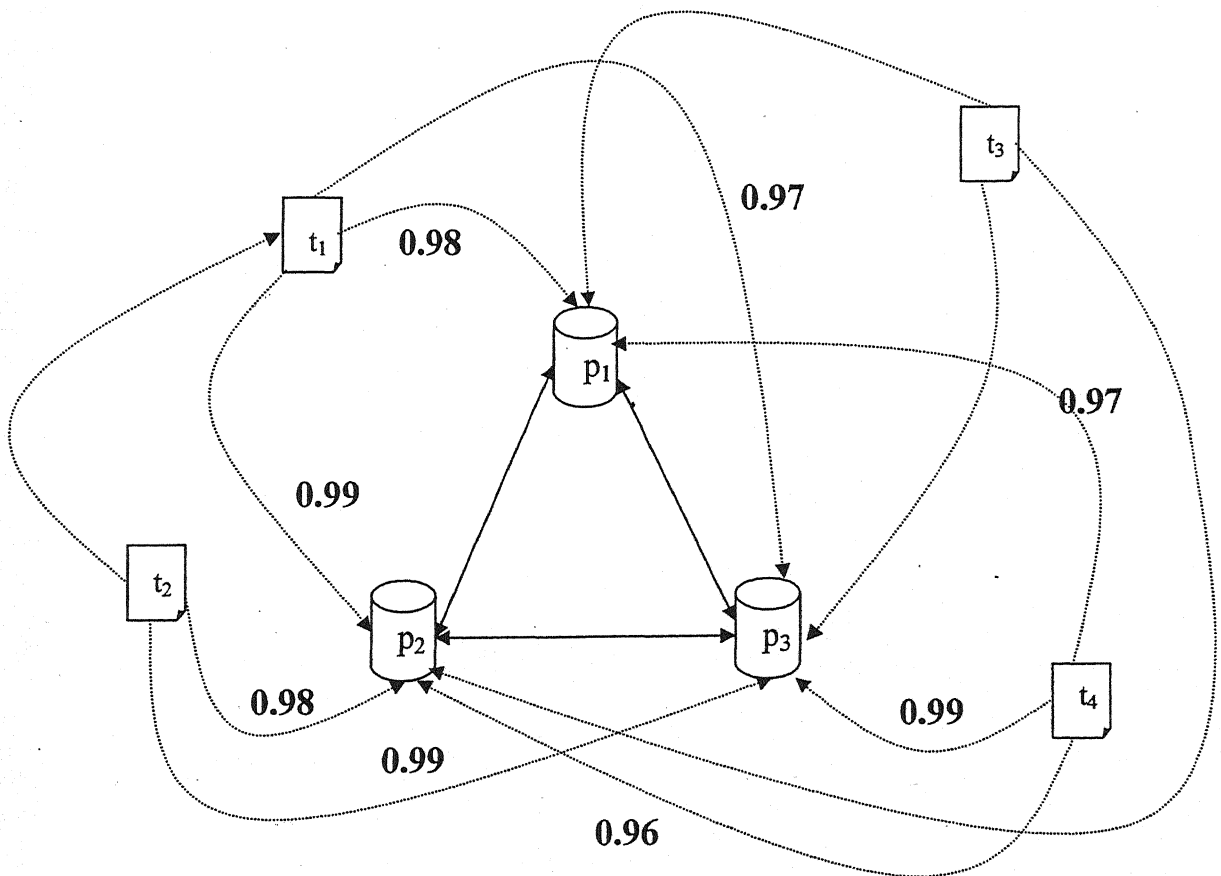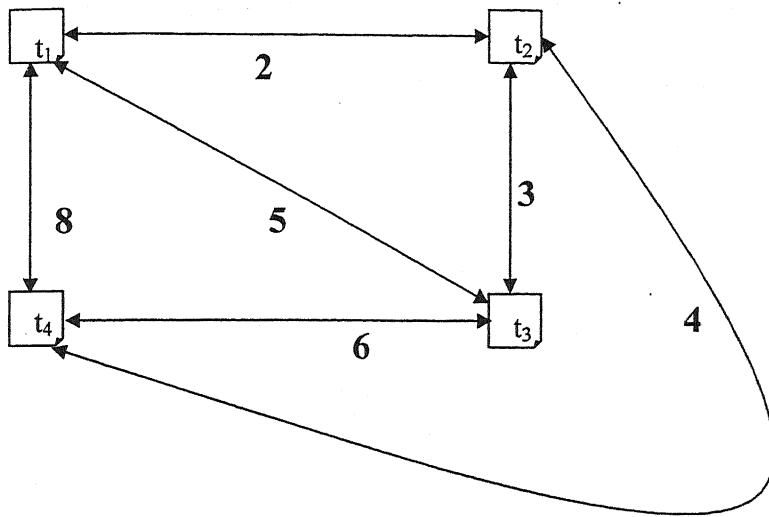| (m, n)	Model [ZAHE 1991] (m^m)	Model $(n)^m$ [KUMA 1999]	Present model (n!)
(4,3)	256	81	6
(5,3)	3125	243	6
(6,3)	46656	729	6
(7,3)	823543	2187	6
(8,3)	16777216	6561	6
(8,4)	16777216	65536	24
(8,5)	16777216	390625	120
(8,6)	16777216	1679616	720
(8,7)	16777216	5764801	5040

Figure 1 Execution Graph

Figure 2 Inter Task Communication Graph
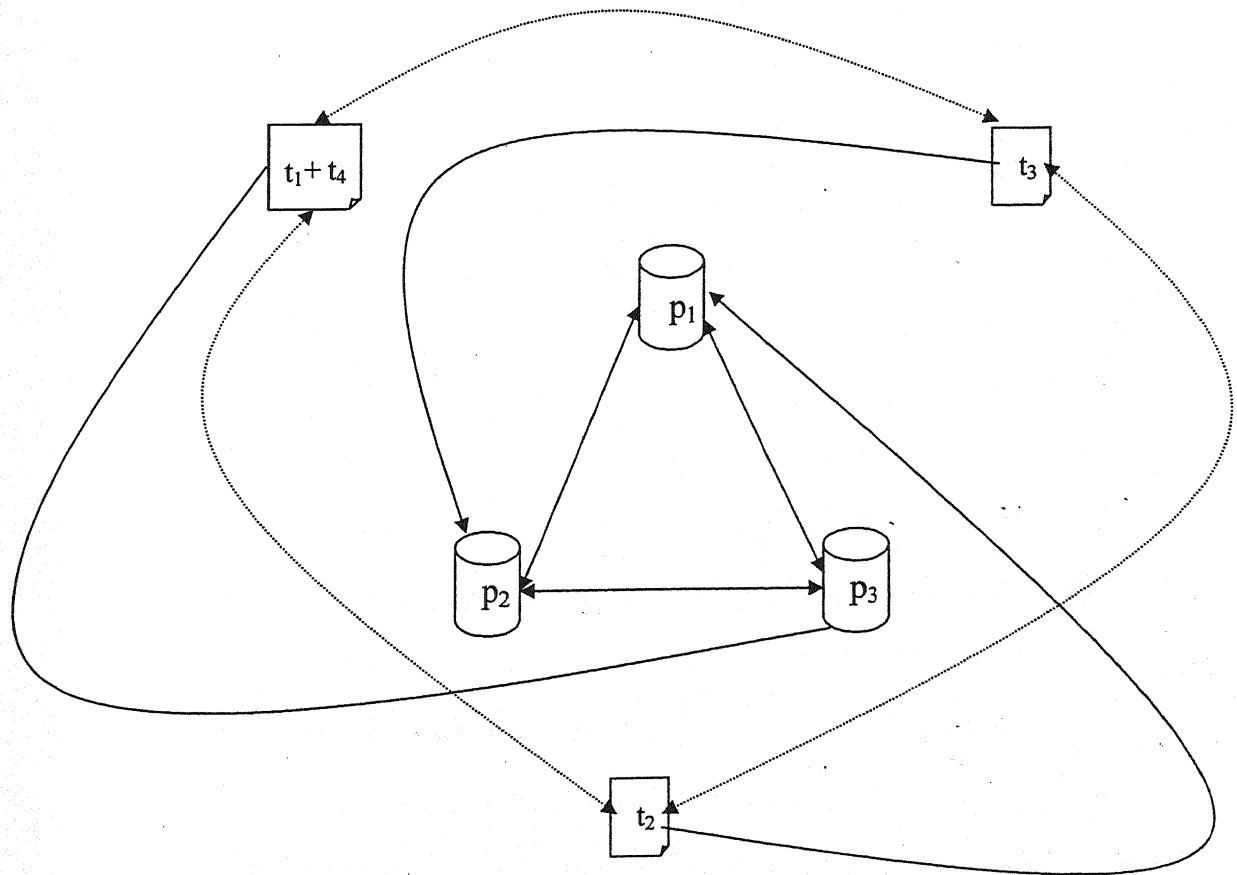
Figure 3 Optimal Assignment Graph

Model-III

AN EFFICIENT ALGORITHM FOR TASKS ALLOCATION TO THE DISTRIBUTED SYSTEMS

3.3.1 Objective

The objective of this research model· is describe as the allocation of the tasks

to the processors in such a way that the overall cost and reliability is to be optimized

under the pre specified constraints. The care has also made to balance the utilization

of each and every processor and to distribute the set of other constraints appropriately.

In order to achieve this objective we shall consider a distributed system and a set of

tasks. Further to fulfill our objective in an optimally way, we have formulated the

functions related to costs and reliabilities. These functions shall be capable to measure

execution and communication cost and also execution and communication

reliabilities. The total execution cost and execution reliability have also obtained.

3.3.2 Technique

To evaluate the optimal cost and reliability for optimal tasks allocation,

initially we have to concentrate on those (m-n) tasks that have the highest probability

of data transfer with the remaining n tasks. Each of these (m-n) tasks (say t_i) is treated

as a candidate to be fused with any one (say t_j) out of the remaining n tasks with

which it has the highest communication cost. The corresponding row and column of

CCM (,) and CRM (,) are then eliminated. Further, all the elements of ith row and j^{th}

row of ECM (,) are added and in the ERM (,), all the elements of ith row and j^{th} row

are to be multiplied together. If any of these entries of ECM (,) are less than infinity,

the above tasks are finally fused. Otherwise, a new task (say t_{j2}) out of the remaining

n-1 tasks with which t_i has the next highest communication cost is then selected and

the above process is repeated till all the (m-n) tasks get fused. On applying this process, all the matrices ECM (,), ERM (,), CCM (,) and CRM (,) get reduce to square matrices of order n. Now the problem remains to determine the optimal cost and reliability of the allocation by considering the task-processing cost and reliability to individual processor(s). To allocate the task to one of the processors, the minimum value of each row and column of ECM (,) is obtained. Let min $\{r_i\}$ represent the minimum row cost value corresponding to the tasks t_i and min $\{c_j\}$ represent the minimum column cost value for processor p_j. These values are then replaced to 0 in ECM (,) and corresponding in ERM (,) also. For allocation purpose a modified version of row and column assignment method of Kumar et al [KUMA 1995c] is employed which allocates a task to a processor where it has minimum execution cost. The overall assignment cost [Ecost] and reliability [Ereliability] is expressed as the sum of execution costs along with communication cost and sum of the products of the execution reliabilities along with communication reliabilities respectively of all the tasks as follows:

$$E \cos t = \left[\sum_{j=1}^{n} \left\{ \sum_{j=1}^{n} EC_{ij} x_{ij} \right\} + \sum_{j=1}^{n} \left\{ \sum_{j=1}^{n} CC_{ij} y_{ij} \right\} \right]$$

$$Ereliability = \left[\prod_{i=1}^{n} \left\{ \sum_{j=1}^{n} ER_{ij} x_{ij} \right\} * \prod_{i=1}^{n} \left\{ \sum_{j=1}^{n} CR_{ij} y_{ij} \right\} \right]$$

where, $x_{ij} = \begin{cases} 1, \text{if } i^{th} \textit{ task is assigned} \text{ to } j^{th} \text{ processor} \\ 0, \text{otherwise} \end{cases}$, and

$y_{ij} = \begin{cases} 1, \text{if the } \textit{task assigned} \text{ to processor i communicate with the task assigned to processor j} \\ 0, \text{otherwise} \end{cases}$

3.3.3 Algorithm

To give an algorithmic representation to the technique as mentioned above, we follow the following steps:

Step-1:

Input: m, n, ECM (,), ERM (,), CCM (,), CRM (,) ; nar : = 0; T_{ass}: = {   } ;

Step-2:

nlp : = m-n

for nl : = 1 to nlp do

begin

for i : = 1 to m do

begin

for j : = 1 to m do

begin

Step-2-1:

Arrange the upper diagonal values of CCM (,) in non ascending order and store the result in MCC (), one dimensional array of order $n' = n(n-1)/2$.

Step-2.2:

Store the corresponding positions in POSCC [i, j], (i = 1,.. ,n' and j = 1,2).

Step-2.3:

Select the tasks corresponding to the maximum value-position, (say t_i, t_k).

Step-2.3.1:

Add k^{th} row of ECM (,) to its ith row. If all the values are become infinite, get next value of MCC ( ) and repeat Step-2.3.

Step-2.3.2:

Multiply k^{th} row of ERM (,) to its ith row. If all the values of the product row are zero, get next value of MCC ( ) , repeat Step-2.3.

Step-2.4:

Modify CCM (,), CRM (,), ECM (,) and ERM (,) as follow;

Step-2.4.1:

Replace the corresponding values of kth row and k^{th} column by zero in CCM (,) and then add k^{th} row to i^{th} row and k^{th} column to i^{th} column, after that delete the kth row and k^{th} column from CCM (,).

Step-2.4.2:

Replace the corresponding values of kth row and k^{th} column by zero in CRM (,) and then add k^{th} row to i^{th} row and k^{th} column to i^{th} column, after that delete the kth row and k^{th} column from CRM (,).

Step-2.4.2:

Modify the ECM (,) by adding k^{th} row to i^{th} row and thereby deleting k^{th} row.

Step-2.4.4:

Modify the ERM (,) by multiplying k^{th} row to i^{th} row and thereby deleting k^{th} row.

If nl ≠ m-n go to Step-2.

Step-2.5:

Store modified, ECM (,), ERM (,), CCM (,) and CRM (,) to NECM (,), NERM (,), NCCM (,) and NCRM (,) respectively.

          end;

     end;

end.

Step-3:

     for k : = 1 to n do

     for j : = 1 to n do

begin

Find the minimum of kth row (Say mr_{kj}) of NECM(,) falling in j^{th} column and replace it by zero and accordingly in NERM (,).

end.

Step-4:

     for k : = 1 to n do

     for j : = 1 to n do

begin

Find the minimum of jth column (say mc_{kj}) of NECM (,), which lies in k^{th} row, and replace it by zero and accordingly in NERM (,).

end.

Step-5:

for j : = 1 to n do

begin

for k : = 1 to n do

begin

Search for a row in NECM (,), which has only one zero say, at the position (k,j) and assign task(s) corresponding to this position.

nar : = nar + 1;

far(k) : = j;

T_{ass} :   = $T_{ass} \cup \{t_k\}$;

end;

end.

Step-6:

for j : = 1 to n do

begin

for k : = 1 to n do

begin

Search for a column which has only one zero entry say, at the position (k,j) and assign task(s) corresponding to this position.

nar : = nar + 1;

far(k) : = j;

T_{ass} :   = $T_{ass} \cup \{t_k\}$;

end;

end.

Step-7:

if nar ≠ n

Pick-up an arbitrary zero entry say, at the position (k,j) and assign task(s) corresponding to this position.

nar : = nar + 1;

far(k) : = j;

T_{ass} : $= T_{ass} \cup \{t_k\}$;

Else

Check column(s) positions of zero(s) in unassigned row (s). Check the row(s) for any previous assignment in the corresponding column(s). Find the minimum of the entire elements for the remaining rows and replace it zero and then go to step-5.

Step-8:

Evaluate Execution Cost and Execution Reliability as;

Step-8.1:

EC : = 0.0;

for i : = 1 to n do

EC : = EC + EC(i, far(i))

Step-8.2:

ER : = 1.0;

for i : = 1 to n do

ER : = ER * ER(i, far(i))

Step-9:

Evaluate Communication Cost and Communication Reliability as;

Step-9.1:

CC : = 0.0;

for i = 1 to n do

CC : = CC + CC(i, far(i))

Step-9.2:

CR : = 1.0;

    for i = 1 to n do

    CR : = CR * CR(i, far(i))

Step-10:

Execution Cost [Ecost] and Execution Reliability [Ereliability] are thus calculated as:

Step-10.1:

Ecost: = EC+ CC

Step-10.2:

Ereliability: = ER * CR

Step-11:

Stop.

3.3.4 Implementation

Example-I

Consider a distributed system consisting of a set $T = \{t_1, t_2, t_3, t_4, t_5\}$ of 5 tasks and a set $P = \{p_1, p_2, p_3\}$ of 3 processors.

Step-1: Input: 5,3,

$$
ECM\,(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array}
\begin{array}{ccc}
p_1 & p_2 & p_3 \\
8 & 12 & 7 \\
9 & 8 & 11 \\
12 & 9 & 6 \\
10 & 11 & 12 \\
7 & 6 & 2
\end{array} ;
$$

$$ERM \, (,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.997 & 0.996 & 0.994 \\ 0.993 & 0.998 & 0.992 \\ 0.998 & 0.991 & 0.994 \\ 0.997 & 0.993 & 0.998 \\ 0.992 & 0.995 & 0.996 \end{array} \; ;$$

$$CCM(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \\ 0 & 3 & 6 & 9 & 8 \\ 3 & 0 & 4 & 5 & 6 \\ 6 & 4 & 0 & 7 & 9 \\ 9 & 5 & 7 & 0 & 5 \\ 8 & 6 & 9 & 5 & 0 \end{array}$$

$$CRM(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \\ 0.000 & 0.994 & 0.996 & 0.995 & 0.993 \\ 0.994 & 0.000 & 0.992 & 0.993 & 0.995 \\ 0.996 & 0.992 & 0.000 & 0.992 & 0.996 \\ 0.995 & 0.993 & 0.992 & 0.000 & 0.992 \\ 0.993 & 0.995 & 0.996 & 0.992 & 0.000 \end{array}$$

Step-2:

Step-2.1 & 2.2 gives the following results:

$$MCC(\,) = \begin{bmatrix} 9 & 9 & 8 & 7 & 6 & 6 & 5 & 5 & 4 & 3 \end{bmatrix}$$

$$POSCC(,) = \begin{array}{cccccccccc} 1 & 3 & 1 & 3 & 1 & 2 & 2 & 4 & 2 & 1 \\ 4 & 5 & 5 & 4 & 3 & 5 & 4 & 5 & 3 & 2 \end{array}$$

Step-2.3:

On applying Step- 2.3, we get:

$$\begin{array}{c} \\ t_1 * t_4 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 18 & 23 & 19 \end{array}$$

Step-2.4:

This step gives the following results:

$$NCCM(,) = \begin{array}{c} \\ t_1 * t_4 \\ t_2 \\ t_3 * t_5 \end{array} \begin{array}{ccc} t_1 * t_4 & t_2 & t_3 * t_5 \\ 0 & 3 & 6 \\ 3 & 0 & 4 \\ 6 & 4 & 0 \end{array}$$

$$NCRM(,) = \begin{array}{c} \\ t_1 * t_4 \\ t_2 \\ t_3 * t_5 \end{array} \begin{array}{ccc} t_1 * t_4 & t_2 & t_3 * t_5 \\ 0.000 & 0.994 & 0.996 \\ 0.994 & 0.000 & 0.992 \\ 0.996 & 0.992 & 0.000 \end{array}$$

$$NECM(,) = \begin{array}{c} \\ t_1 * t_4 \\ t_2 \\ t_3 * t_5 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 18 & 23 & 19 \\ 9 & 8 & 11 \\ 19 & 15 & 8 \end{array}$$

$$NERM(,) = \begin{array}{c} \\ t_1 * t_4 \\ t_2 \\ t_3 * t_5 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.994009 & 0.989028 & 0.992012 \\ 0.993 & 0.998 & 0.992 \\ 0.990016 & 0.986045 & 0.990024 \end{array}$$

Step-3 & 4

Here, min $\{r_i\}$ from NECM (,) for every i are $r_{11} = 18$, $r_{22} = 8$, $r_{33} = 8$

Making, $r_{11} = r_{22} = r_{33} = 0$ and accordingly in NERM (,) also, we gets;

$$NECM \;(,) = \begin{array}{c} \\ t_1 * t_4 \\ t_2 \\ t_3 * t_5 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0 & 23 & 19 \\ 9 & 0 & 11 \\ 19 & 15 & 0 \end{array}$$

$$NERM \;(,) = \begin{array}{c} \\ t_1 * t_4 \\ t_2 \\ t_3 * t_5 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0 & 0.989028 & 0.992012 \\ 0.993 & 0 & 0.992 \\ 0.990016 & 0.986045 & 0 \end{array}$$

Step-5, 6 & 7:

The following allocations are obtained after implementing the row & column assignment process [KUMA 1995c];

83

Tasks	→	Processors	EC	ER	CC	CR
$t_1 * t_4$	→	p_1	18	0.994009	3	0.994
t_2	→	p_2	8	0.998	6	0.996
$t_3 * t_5$	→	p_3	8	0.990024	4	0.992

Step-8:

$EC := 34$

$ER := 0.9821245$

Step-9:

$CC := 13$

$CR := 0.9821038$

Step-10:

Ecost   :=   47

Ereliability   :=   0.9645482

Step-11:

Stop.

Example-II

Consider another distributed system which have a set of 10 tasks $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}\}$ and a set of 5 processors $P = \{p_1, p_2, p_3, p_4, p_5\}$.

$$ECM(,) = \begin{array}{c|ccccc} & p_1 & p_2 & p_3 & p_4 & p_5 \\ t_1 & 5 & 3 & 2 & 7 & 8 \\ t_2 & 11 & 13 & 3 & 5 & 7 \\ t_3 & 14 & 10 & 8 & 2 & 5 \\ t_4 & 11 & 3 & 2 & 7 & 9 \\ t_5 & 12 & 15 & 3 & 8 & 4 \\ t_6 & 2 & 6 & 7 & 5 & 8 \\ t_7 & 7 & 9 & 8 & 2 & 4 \\ t_8 & 15 & 9 & 11 & 6 & 3 \\ t_9 & 7 & 5 & 3 & 8 & 12 \\ t_{10} & 2 & 4 & 2 & 13 & 15 \end{array}$$

$$ERM(,) = \begin{array}{c|ccccc} & p_1 & p_2 & p_3 & p_4 & p_5 \\ t_1 & 0.997 & 0.996 & 0.993 & 0.991 & 0.994 \\ t_2 & 0.992 & 0.995 & 0.991 & 0.997 & 0.992 \\ t_3 & 0.998 & 0.991 & 0.995 & 0.997 & 0.999 \\ t_4 & 0.995 & 0.998 & 0.991 & 0.994 & 0.998 \\ t_5 & 0.992 & 0.997 & 0.993 & 0.995 & 0.994 \\ t_6 & 0.998 & 0.999 & 0.992 & 0.991 & 0.997 \\ t_7 & 0.994 & 0.995 & 0.998 & 0.999 & 0.992 \\ t_8 & 0.991 & 0.996 & 0.999 & 0.997 & 0.995 \\ t_9 & 0.992 & 0.995 & 0.991 & 0.994 & 0.998 \\ t_{10} & 0.991 & 0.993 & 0.995 & 0.997 & 0.999 \end{array}$$

$$CCM(,) = \begin{array}{c|cccccccccc} & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} \\ t_1 & 0 & 3 & 6 & 9 & 0 & 1 & 2 & 0 & 2 & 4 \\ t_2 & 3 & 0 & 4 & 5 & 3 & 0 & 5 & 9 & 0 & 8 \\ t_3 & 6 & 4 & 0 & 7 & 0 & 8 & 3 & 0 & 2 & 0 \\ t_4 & 9 & 5 & 7 & 0 & 9 & 7 & 0 & 2 & 0 & 1 \\ t_5 & 0 & 3 & 0 & 9 & 0 & 6 & 3 & 1 & 0 & 0 \\ t_6 & 1 & 0 & 8 & 7 & 6 & 0 & 8 & 0 & 7 & 9 \\ t_7 & 2 & 5 & 3 & 0 & 3 & 8 & 0 & 0 & 6 & 5 \\ t_8 & 0 & 9 & 0 & 2 & 1 & 0 & 0 & 0 & 5 & 0 \\ t_9 & 2 & 0 & 2 & 0 & 0 & 7 & 6 & 5 & 0 & 1 \\ t_{10} & 4 & 8 & 0 & 1 & 0 & 9 & 5 & 0 & 1 & 0 \end{array}$$

$$CRM(,) = \begin{array}{c|cccccccccc} & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} \\ \hline t_1 & 0.000 & 0.991 & 0.997 & 0.992 & 0.998 & 0.995 & 0.993 & 0.992 & 0.991 & 0.999 \\ t_2 & 0.991 & 0.000 & 0.992 & 0.000 & 0.994 & 0.991 & 0.998 & 0.996 & 0.000 & 0.991 \\ t_3 & 0.997 & 0.992 & 0.000 & 0.991 & 0.994 & 0.996 & 0.000 & 0.998 & 0.996 & 0.994 \\ t_4 & 0.992 & 0.000 & 0.991 & 0.000 & 0.998 & 0.992 & 0.996 & 0.999 & 0.991 & 0.000 \\ t_5 & 0.998 & 0.994 & 0.994 & 0.998 & 0.000 & 0.999 & 0.991 & 0.994 & 0.000 & 0.995 \\ t_6 & 0.995 & 0.991 & 0.996 & 0.992 & 0.999 & 0.000 & 0.995 & 0.994 & 0.991 & 0.992 \\ t_7 & 0.993 & 0.998 & 0.000 & 0.996 & 0.991 & 0.995 & 0.000 & 0.991 & 0.998 & 0.995 \\ t_8 & 0.992 & 0.996 & 0.998 & 0.999 & 0.994 & 0.994 & 0.991 & 0.000 & 0.998 & 0.991 \\ t_9 & 0.991 & 0.000 & 0.996 & 0.991 & 0.000 & 0.991 & 0.998 & 0.998 & 0.000 & 0.997 \\ t_{10} & 0.999 & 0.991 & 0.994 & 0.000 & 0.995 & 0.992 & 0.995 & 0.991 & 0.997 & 0.000 \end{array}$$

The following allocations and optimal results are obtained as mentioned below, after implementing the suggested algorithm-5.4.3:

Tasks	→	Processors	EC	CC	ER	CR	E cost	Ereliability
$t_6 * t_{10}$	→	p_1						
$t_1 * t_4$	→	p_2						
$t_7 * t_9$	→	p_3	41	40	0.9597109	0.94916	81	0.9109191
$t_2 * t_8$	→	p_4						
$t_3 * t_5$	→	p_5						

3.3.5 Conclusion

The model addressed here is based on the consideration of execution cost and execution reliability of the system and also the communication cost and communication reliability among the tasks are taken into account for the purpose. The method is presented in algorithmic form and implemented on the several sets of input data to test the performance and effectiveness of the algorithm. The graphical representation of the distributed computing system taken in to consideration for the said purpose shown through graph 1 to 2, in the graph 1 the execution costs and

reliabilities of each tasks to each processors has been shown while in graph 2, shows the communication costs and reliabilities amongst the tasks. After implementing the proposed technique on this tasks allocation problem a modified version of execution and communication assignment graph have been obtained which are depicted to graph 3 and 4. On the basis of the modified effectiveness matrices as obtained in step-2 of the algorithm and after implementing the modified version of row and column assignment method devised by Kumar et al [KUMA 1995c] the following table shows the results as obtain after implementing the present algorithm.

Tasks $\rightarrow$ Processors	EC	ER	CC	CR	E cost	E reliabilty
$t_1 * t_4 \rightarrow p_1$						
$t_2 \rightarrow p_2$	34	0.9821245	13	0.9821038	47	0.9645482
$t_3 * t_5 \rightarrow p_3$						

The graphical representation is given in the graph 5, it is clear from the graph that tasks t_1 and t_4 goes to p_1, task t_2 goes to p_2 and tasks t_3 and t_5 goes to p_3. The total optimal cost and the total optimal reliability have been obtained which are 47 and 0.9645482 respectively. The run time complexity of the algorithm is measured O $(6mn-n^2)$ time. Here it is concluded that it remains same as [KUMA 1998, YADA 2002]. Our time complexity is better than O $(m^2 n)$ of [SAGA 1991] and O (n^m) to [PENG 1997, RICH 1982]. The performance of the algorithm is compared with that of [SAGA 1991] and [PENG 1997, RICH 1982] and result in both cases, i.e. when tasks increases against the fixed processors and the case in which the tasks remain fix, while increase in processors are shown below:

Case-1: For fixed processors n = 3

Taks m	$O(6mn - n^2)$ Present $-A$ lg	$O(m^2 n)$ [SAGA1991]	$O(n^m)$ [PENG1997, RICH1982]
5	81	75	243
6	99	108	729
7	117	147	2187
8	135	192	6561
9	153	243	19683
10	171	300	59049

Case-2: For fixed tasks m = 10

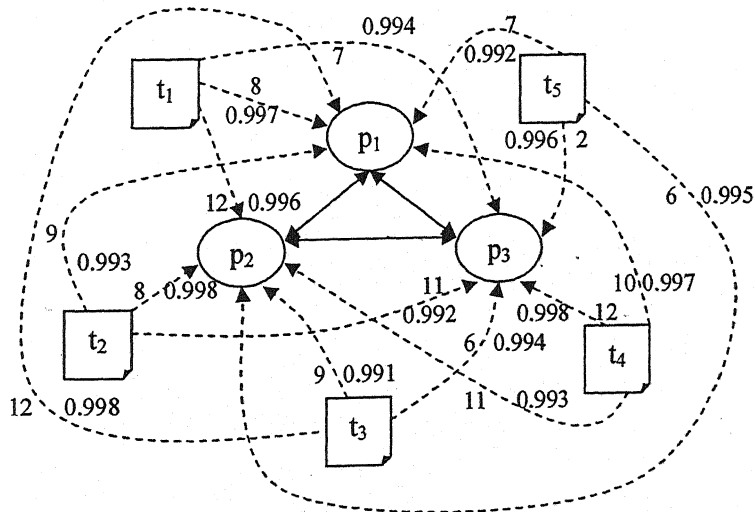| Processors n	$O(6mn - n^2)$ Present $-A$ lg	$O(m^2 n)$ [SAGA1991]	$O(n^m)$ [PENG1997, RICH1982]
3	171	300	59049
4	224	400	1048576
5	275	500	9765625
6	324	600	60466176
7	371	700	282475249
8	416	800	1073741824

Figure 1: Combined Graphs for Execution Cost Matrix And Execution Reliability Matrix
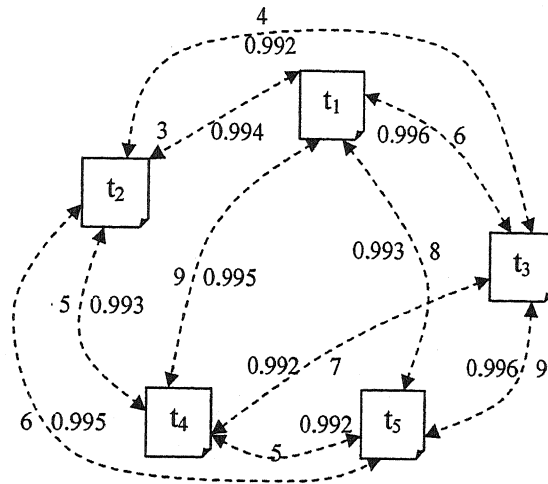
Figure 2:   Combined Graphs for Communication Cost Matrix
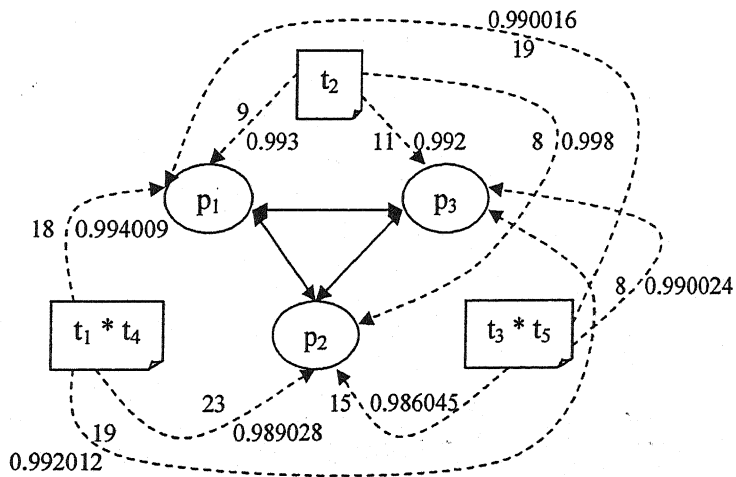   And Communication Reliability Matrix

Figure 3:   Combined Graphs for Modified Execution Cost Matrix
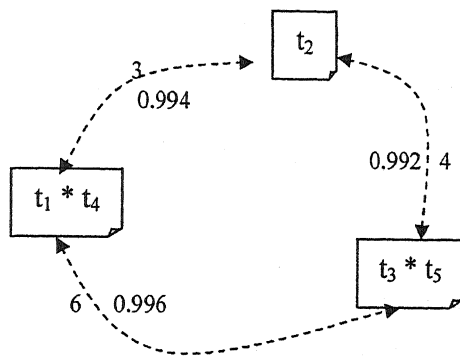   and Modified Execution Reliability Matrix

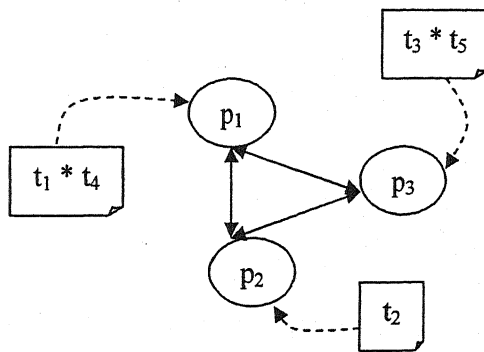Figure 4: Modified Communication Cost and Modified Communication Reliability Matrix

Figure 5:   Optimal Assignment Graph

MODEL-IV

COMMUNICATION REDUCTION BASED ALGORITHM FOR OPTIMAL ASSIGNMENT IN DISTRIBUTED SYSTEMS

3.4.1 Objective

Let the given distributed system consists of a set of n – processor, p = [p_1 , p_2---p_n] interconnected by communication links. These links serve the purpose of transferring messages / data between the processors, and a set of m tasks T = [t_1, t_2----t_m] that constitute a communicated program. These tasks are collectively responsible for attaining the desired goal. The processing efficiency of individual processor is given in the form of matrix ECM (,) of order m x n and ITCC is taken in the form of a symmetric a matrix CCM (,) of order m x m. The proposed model relies upon:

- Suggesting way to assigning tasks which have maximum ITCC.

- Formulating the cost function to measure EC,

- Formulating the cost function to measure ITCC,

- Deriving a procedure to assign the tasks, and

- Developing an algorithm to obtain optimal cost.

3.4.2 Technique

A task is allocated to a processor in such a way that extensive inter task communication is avoided and the capabilities of the processor suit to the execution requirements of the task. The proposed allocation policy involves stepwise assignment. To begin, first we concentrate on task-pair having maximum communication. Let such a task pair be (t_l, k_k). We find the communication of t_i with all tasks and t_k with all tasks. Check both pair of task having maximum communication then assigned them on same processor.

91

Now check again, the maximum communication of next unassigned task with all the assigned tasks and which has maximum then assign them on same processor till the total assignments are not completed. With the help of total assignments is ECM (,) calculate the total execution cost (TEC) and total communication cost (TCC) and lastly we calculate the total optimal cost i.e. TOC,

$$TOC = TEC + TCC$$

TEC, the total execution cost and TCC, the total communication cost are given by

$$TEC = \sum_{i=1}^{m} \sum_{j=1}^{n} e_{ij} \, x_{ij}$$

$$TCC = \sum_{i=1}^{m-1} \sum_{k>i}^{n} c_{ik}$$

Where,

$$x_{ij} = \begin{cases} 1 \text{ if } i^{th} \text{ task is assigned to } j^{th} \text{ processor} \\ 0, \text{ otherwise} \end{cases}$$

The above technique can be described in the form of an algorithm as given below.

3.4.3 Algorithm

Step-1: Input: m,n;//m is the number of tasks, n is the number of processors//

Input: PI()//linear array to hold the processors number//

Input: ECM(,); // matrix to hold the execution cost of each task on each processor//

Input:// ITCCM (,); //matrix to hold the communication cost between the tasks//

Step-2:   maxcc( ) ← 0;// linear array to store the inter task communication costs in decending order.//

minmaxcc()←0://linear array to hold the minimum of maxcc()//

bmin ← 0;// variable for selecting the minimum //

T_{ass}( ) ← 0; // linear array to hold the tasks in order of assignment made //

$T_{non-ass}$( ) ← 0; //linear array to store the non assigned tasks//

nomade ← 0; // variable for counting the number of assignments has been made//

nccm()←;//operational matrix for execution cost//

allock(,) ← 0; //an array to hold processors' positions and related assign tasks//

pallock()←0;//linear array to hold the order of processor in which assigne next made//

trp ( ) ← 0; // linear array to store the throughput of the processors//

tempcc()←0;//store the sum of EC//

POS (,) ← 0; // two dimensional array to hold to corresponding position of maxcc ( )//

ncount←0;//variable counter//

CPOS (,) ← 0; // two dimensional array to hold the positions of common elemenst//

IPOS(,)←0;//an array to hold the position for minmaxcc()//

ttask ( ) ← 0; //linear array to store the total number of tasks to the processors//

count←0;//counter//

Step-3:   set k ←m (m-1)/2

set t ← m-n

Step-4:   for i← 1 to m do

for j ← 1 to k do

arrange the above diagonal value of ITCCM(,) in decending order

and store the result in maxcc( ) unitl j=k,

repeat

repeat

Step-4.1:   for i ← 1 to k do

for j ← 1 to 2 do

store the corresponding position of maxcc( ) in POS(,)

repeat

    repeat

Step-4.1.1:   set count $\leftarrow$ n

    set ncount$\leftarrow$ 0

        for i $\leftarrow$ 1 to m do

            for j $\leftarrow$ 1 to k do

    pick –up the first minimum from maxcc () (say $maxcc_j$)

    minmaxcc $\leftarrow$ $maxcc_j$

    compare with other value of maxcc ( )

    if $maxcc_j \neq$ maxcc ( )

    then

        Pick-up the correspond my position form pos (,) and store the result in IPOS (,)

    ncount $\leftarrow$ ncount +1

    else

        pick-up the next minimum from maxcc at j^{th} position say $(maxcc_j)$

    minmaxcc ()$\leftarrow$ $maxcc_j$

    pick-up the correspondign postion form POS(,) and

    store the resultin IPOS(,)

        ncount $\leftarrow$ count +1

     endif

     repeat

  repeat

Step-4.1.1.1: for i $\leftarrow$ 1to ncount do

    modify the POS(,)by deleting the position in IPOS(,) and reduce the maxcc () by elminating the value stored in minmaxcc ( )

    for k $\leftarrow$ 1 to m do

    for j $\leftarrow$ 1 to n do

        store the ECM(,)in NCCM(,)

            NCCM(,)$\leftarrow$CCM (,)

    repeat

  repeat

for k $\leftarrow$ 1 to mcount

for j $\leftarrow$ 1 to n

modify the NECM (,) by eleminating the positions stored in POS (,) and assign the tasks/pair as per there minimum Σe_{ij} and store the volve in TEMPEC (,)

repeat

    repeat

    for i ← 1 to ncount do

for j ← 1 to 2 do

    select the minimum of TEMPEC (,) say $Tempe_i$, pick-up the position form I POS (,) related to $tempe_i$

repeat

repeat

Step-4.1.1.2: for i ← 1 to count do

      assign t_j to p_i

      allock (i,j) ← r

    pallock()←r

    nomade = nomade +1

    T_{ass} ← T_{ass} U (t_j)

repeat

Step-4.1.1.3: if n = count

      then go to step 5

    else

if POS (i,j) < POS (i,k)

bmin ← POS (i,j) until j = k repeat for i

    bmin < POS (i, j) until j=k.

  endif

    check the corresponding position of bmin in POS (,) and give one increment to count as:

for j ←1 to n do

    check the processor which is not belongs to Pallock ( )

    as p_i( ) U Pallock ( )say p_j assign tasks t_l to p_j

allock (l,j) ← j

    nomade ← nomade +1

count ← count +1

endif

store the remaining (m-n) tasks in a linear array $T_{non\text{-}ass}$ ( )

Step-5:

for i= 1 to m do

for j= 1 to n do.

Select a task laying in $T_{non-ass}$ (,) say t_j and check the maximum communication with assign task laying in T_{ass}(,), say t_k.

Then check the EC.

If it has finite EC then assign then task to same processor and store it on T_{ass} (,) and remove the task from $T_{non-ass}$ ( ).

This process will continue all the tasks in $T_{non-ass}$ ( ). get assigned.

If $= \infty$ then select the next maximum communication cost and assign them.

    Until $T_{non-ass} = [\quad]$.

     repeat

repeat

Step-6:

TEC: = O;

for i= 1 to m do

for j= 1 to n do.

Compute the Total Execution Cost as;

TEC = TEC + ECM (i,j);

    repeat

repeat

Step-6.1:

TCC: = O;

for i:= 1 to m do

for j:= 1 to m do.

Compute the Total ITCC as;

TCC: = TCC + ITCCM (i,j);

    repeat

repeat

Step-6.2:

Compute the total system cost by summing up the

TEC and TCC as;

TOC:= TEC +TCC;

Step-7:

Stop

3.4.4 Implementation

Step-1:

Input m =8, n=2, and

ECM (,) =

	p_1	p_2
t_1	8	5
t_2	4	1
t_3	6	2
t_4	7	9
t_5	2	7
t_6	1	3
t_7	2	4
t_8	5	1

and

CCM (,)=

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
t_1	0	1	4	3	2	5	1	0
t_2	1	0	0	0	0	0	0	0
t_3	4	0	0	1	5	7	0	0
t_4	3	0	1	0	2	6	1	5
t_5	2	0	5	2	0	0	0	4
t_6	5	0	7	6	0	0	6	8
t_7	1	0	0	1	0	6	0	3
t_8	0	0	0	5	4	8	3	0

Step-2:

Arrange the above diagonal values of CCM (,) in descending order and store the result in a linear array ICC ( ).

ICC ( ) =   {8,7,6,6,5,5,5,4,4,3,3,2,2,1,1,1,1,0,0,0,0, 0,0,0,0,0,0,0}.

Step-3:

Store the corresponding position of the results in POS (i,j) where

i = 1,2........m (m-1)/2

j = 1,2.

POS (i,j) =   [(6,8), (3,6),(4,6), (6,7), (1,6), (3,5), (4,8), (1,3), (5,8), (1,4), (7,8), (1,5), (4,5), (1,2), (1,7), (3,4), (4,7), (1,8), (2,3), (2,4), (2,5), (2,6), (2,7), (2,8), (3,7), (3,8), (5,6), (5,7)].

Step-4:

Check minimum communication cost of the two tasks in CCM (,),

Here minimum communication

i.e. bmin = between the tasks t_6 and t_8=0.

Than assign these two tasks to the processors.

Step-5:

Assigned and non-assigned tasks are,

T_{ass} = [t_6 , t_8]

alloc : [1, 2],

and $T_{non-ass}$ = [t_1,t_2, t_3, t_4, t_5, t_7]

Step-6:

for i= 1 to m do

for j= 1 to n do

check the value of alloc for t_6 say p_j then assing the task t_k to p_j

    repeat

repeat

Step-6.1:

Select the task t_1 in $T_{non-ass}$ ( )and check maximum communication with the assigned tasks i.e. t_6, t_8.

Maximum communication of t_1 with t_6 and t_8 is,

$[t_1, t_6] = 5$

Then assign t_1 to the processor p_1. Therefore,

$$T_{ass} = [t_6, t_8, t_1]$$

and $T_{non-ass} = [t_2, t_3, t_4, t_5, t_7]$

Step-6.2:

Repeating the step 6.1 to 6.6 we get the following assignments:

ECM (,) =

	P_1	p_2
t_1	⑧	5
t_2	④	1
t_3	⑥	2
t_4	⑦″	9
t_5	②	7
t_6	①	3
t_7	②	4
t_8	5	①

Step-7:

The total execution cost of the assignments made in previous step is 30 and total communication cost is 20.

Step-7.3:

Total optimal system cost = TEC + TCC

TOC = 30 + 20 = 50

Step-8:

End

3.4.5 Conclusion

Here we have presented a model to assign the tasks to the processors by reduction based technique. One can see that present model gives better results which are

depicted in the table 4.1. The graphical representation has been shown through figure

3.4.1. to 3.4.4.

Table – 4.1

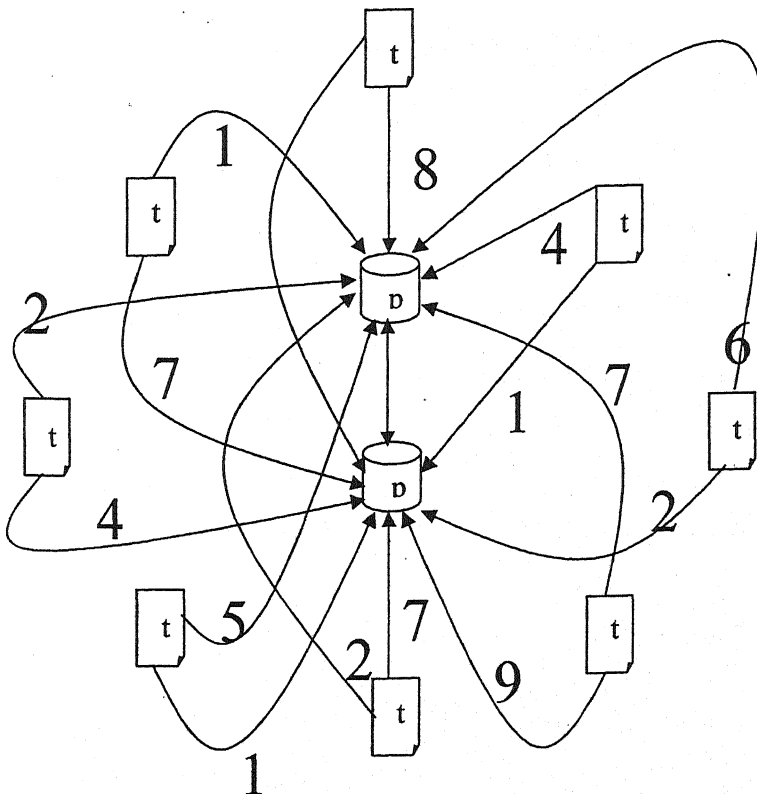|	Total optimal system cost	
Present Method	50	36
[Yada 95]	50	23

Figure 1 Processor Connection Graph
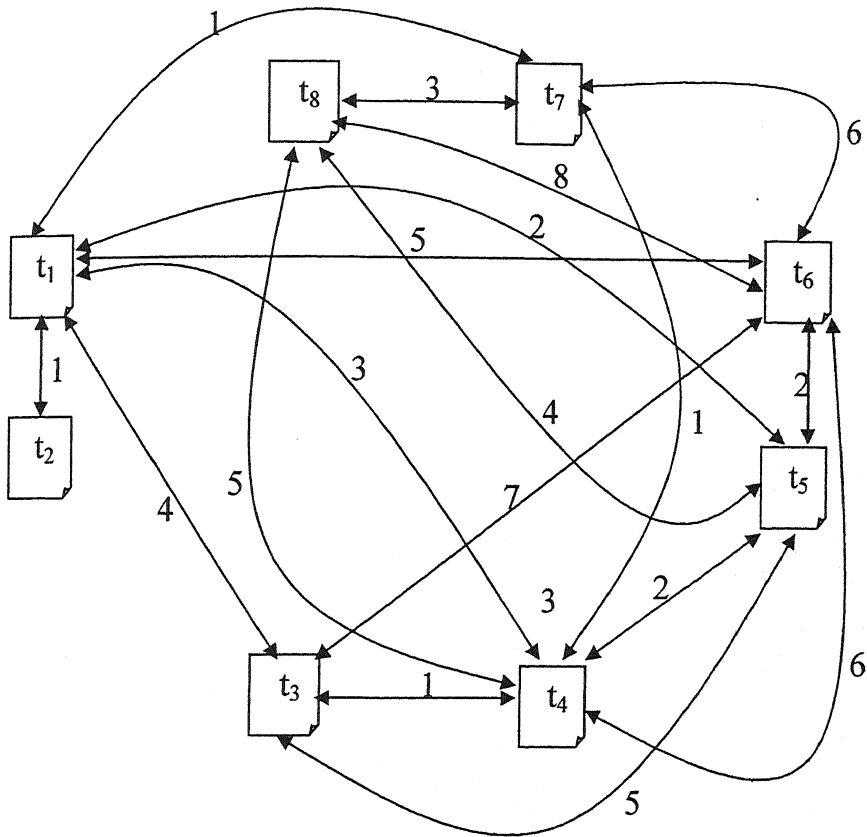

Figure 2 Tasks Execution Graph

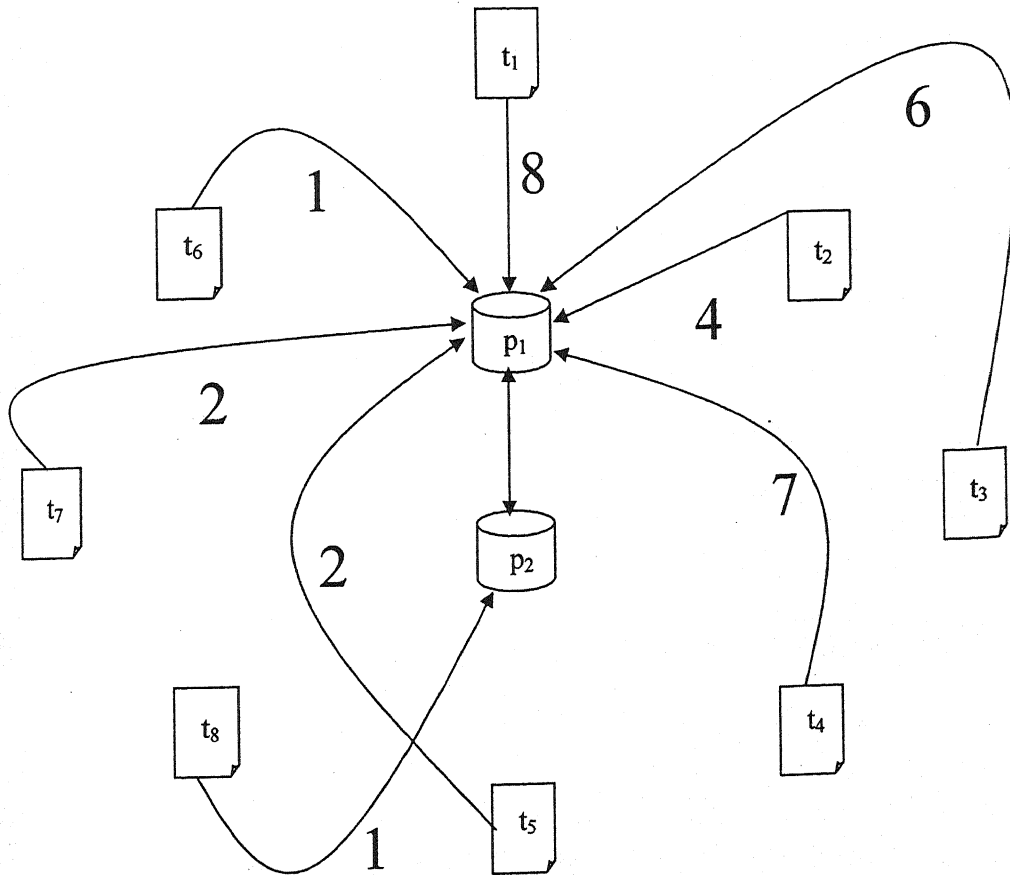Figure 3 Inter Task Communication Graph

Figure 4 Optimal Assignment Graph

MODEL-V

OPTIMIZING THE RELIABILITY OF DISTRIBUTED SYSTEM

3.5.1 Objective

A procedure to assign the tasks to the processors of the Distributed Computing systems based on execution reliability is to be designed in such a way that the overall reliability is to be optimizing under the pre specified constraints and non of the tasks get unexecuted. The reliability functions to measure ER and CR are then formulated.

3.5.2 Technique

Since the number of tasks are more than the number of processors, so that we divide the problem in to sub balance problems. First of all obtain the product of each row and each column except, the position where, the reliability is zero (zero reliability should be kept aside with the product of row or column) from the ERM(,) and store the results in to Product_Row() and Prouct_Column(), each of them are one dimensional arrays. Select the first set of tasks, (this set of tasks shall contain only as many tasks as the number of processor) on the basis of maximum reliability against the tasks in the Product_Row() array. Store the result in to ERM (, ,) a two dimensional array. Repeat the process until remaining tasks are either less than or equal to the number of processors. If the tasks are equal to the processors of then it will become the last sub problem, else to form the last problem we have to delete the column (processor) from ERM(, ,) on the basis of Product_Column() array i.e. this set shall contain only as many processors as number of tasks left, so that we delete the processors have lesser reliability in the Product_Column() array. Convert matrices of each of sub problems in to execution unreliability matrix namely, EURM(, ,) by

subtracting each element of ERM(, ,) from one. For allocation purpose a modified version of row and column assignment method devised by Kumar et al [KUMA 1995c] is employed which allocates a task to a processor where it has minimum execution unreliability. The communication unreliability of those tasks, which are allocated on the same processor, becomes one. For each sub problem we calculate the exaction reliability of each processor and store the result in a linear array PER(j) and also communication reliability PCR(j) of the distributed computing system, where j= 1,2,...n. The overall assignment reliability [Ereliability] is expressed as products of the execution reliabilities and communication reliabilities of all the tasks as follows:

$$PER(j) = \prod_{i=1}^{n} \left\{ \sum_{j=1}^{n} er_{ij} x_{ij} \right\} ;$$

$$PCR(j) = \prod_{i=1}^{m} \left\{ \sum_{j=1}^{n} cr_{ij} y_{ij} \right\}$$

$$Ereliabili\ ty = \left[\prod_{i=1}^{n} \left\{ \sum_{j=1}^{n} er_{ij} x_{ij} \right\} * \prod_{i=1}^{n} \left\{ \sum_{j=1}^{n} cr_{ij} y_{ij} \right\} \right]$$

where, $x_{ij} = \begin{cases} 1, \text{ if } i^{th} \text{ } task \text{ } is \text{ } assigned \text{ to } j^{th} \text{ processor} \\ 0, \text{ otherwise} \end{cases}$, and

$y_{ij} = \begin{cases} 1, \text{ if the } task \text{ } assigned \text{ to processor i communicate with the task assigned to processor j} \\ 0, \text{ otherwise} \end{cases}$

3.5.3 Algorithm

Step-1:

Input: m, n. ERM (,), CRM(,)

Step-2:

Obtain the product of each row of the ERM(,) in such a way that, if any reliability(ies) is (are) zero then keeping it aside along with sum of that row (just to avoid the condition sum ER * 0 = 0). Store the results in one-dimensional array Product _Row (,) of order m.

Step-3:

Obtain the product of each column of the ERM (,), in such a way that if any reliability (ies) is (are) zero then keeping it aside along with product of that column (just to avoid the condition ER * 0 = 0). Store the results in one-dimensional array Product _Column (,) of order n.

Step-4:

Partitioned the execution reliability matrix ERM (,) of order m x n to sub matrices such that the order of these matrices become square i.e. number of row should be equal to number of column. Partitioning to be made as mentioned in the following steps.

Step-4.1:

Select the n task on basis of Product _Row (,) array i.e. select the 'n' task corresponding to most maximum product to next maximum product, if there is a tie select arbitrarily. (For the cases in which product is ER * 0 = 0, maximum value depends only ER and the impact of zero is to be neglected).

Step-4.2:

Store the result in the two dimensional array ERM (, ,) to form the sub matrices of the sub problems.

Step-4.3:

If all the tasks are selected then go to step 4.7 else steps 4.4

Step-4.4:

Repeat the step 4.1 to 4.3 until the number of task become less than n.

Step-4.5:

Select the remaining task say r, r < n, select the r processors on the basis of Product_ Column (,) array i.e. the processors corresponding to the most maximum product to next maximum, if there is a tie select arbitrarily (for the cases in which product is ER * 0 = 0, maximum value depend only ER and the impact of zero is to be neglected).

Step-4.6:

Store the result in the two dimensional array ERM (, ,), which is a last sub problems.

Step-4.7:

List of all the sub problems formed through Step 4.1 to 4.6 and repeat step 5 to step 13 to solve each of these sub problems.

Step-4.8:

Convert all execution reliability matrices of each sub problems in to the execution unreliability matrices by subtracting each of entry by one in all execution reliability matrices.

Step-5:

Find the minimum of each row of NEURMI (, ,), and replace it by 0.

Step-6:

Find the minimum of each column of NEURM(, ,), and replace it by 0.

Step-7:

Search for a row in NEURM, which has only one zero and assign the task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position.

Step-8:

Search for a column in NEURM, which has only one zero and assign the task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position.

Step-9:

Check whether nar = n if not than pickup an arbitrary 0 and assigned task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position, else, Check column (s) position of 0's in unassigned row(s). Check the row(s) for any previous assignment in the corresponding column(s). Find the minimum of the entire elements for the remaining rows and replace it zero.

Step-10:

Evaluate Execution Reliability.

Step-11:

Evaluate Communication Reliability.

Step-12:

Execution Reliability (Ereliability) are thus calculated as:

Ereliability = ER * CR

Step-13:

Stop.

3.5.4 Implementation

Example-I

Consider a distributed system consisting of a set $T = \{t_1, t_2, t_3, t_4\}$ of 4 tasks and a set $P = \{p_1, p_2, p_3\}$ of 3 processors,

Step-1: Input: 4, 3

$$ERM\ (,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.997 & 0.996 & 0.994 \\ 0.993 & 0.998 & 0.992\ ; \\ 0.998 & 0.991 & 0.994 \\ 0.997 & 0.993 & 0.998 \end{array}$$

$$CRM\ (,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \\ 0.000 & 0.994 & 0.996 & 0.995 \\ 0.994 & 0.000 & 0.992 & 0.993 \\ 0.996 & 0.992 & 0.000 & 0.992 \\ 0.995 & 0.993 & 0.992 & 0.000 \end{array}$$

Step-2:

Obtain the product of each row and column of ERM (,), i .e. the products of each row and each column are as:

$$
Product\_Row() = \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \\ 0.9870539 & 0.9830858 & 0.9830838 & 0.9880409 \end{array}
$$

$$
Product\_Column() = \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.9850768 & 0.9781664 & 0.9781714 \end{array}
$$

Step-3:

We partitioned the matrix ERM (,) to define the first sub problem ERMI (, ,) by selecting rows corresponding t_1, t_2, t_4 on the basis of Product_Row() array and on the basis of the Product_Column() array, by deleting columns corresponding to p_2, p_3 then after selecting the remaining one task t_3 to form the last sub problem ERMII (, ,), as there was only one task, for which we required only one processor. So that the modified matrices are as;

Sub Problem-I:

$$
ERMI(,,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_1 & 0.997 & 0.996 & 0.994 \\ t_2 & 0.993 & 0.998 & 0.992 \\ t_4 & 0.997 & 0.993 & 0.998 \end{array}
$$

and, Sub Problem-II:

$$
ERMII(,,) \begin{array}{c|c} & p_1 \\ t_3 & 0.998 \end{array}
$$

Obtain the unreliability matrices for the ERMI (,,) and ERMII (,,) as,

$$
EURMI(,,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_1 & 0.003 & 0.004 & 0.006 \\ t_2 & 0.007 & 0.002 & 0.008 \\ t_4 & 0.003 & 0.007 & 0.002 \end{array}
$$

$$
EURMII(,,) \begin{array}{c|c} & p_1 \\ t_3 & 0.002 \end{array}
$$

Step-4 & 5:

On applying modified Hungarian method devised by Kumar et al [KUMA 1995c] to assign the tasks, on the basis of min $\{r_i\}$ and min $\{c_j\}$ from unreliability matrices for every i and j. We put $r_{ij} = 0$ and $c_{ij} = 0$, for every i and j. On applying this to all sub problems, the modified matrices for each sub problem are mentioned below:

$$EURMI(,,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_4 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.000 & 0.004 & 0.006 \\ 0.007 & 0.000 & 0.008 \\ 0.003 & 0.007 & 0.000 \end{array}$$

$$EURMII(,,) \begin{array}{c} \\ t_3 \end{array} \begin{array}{c} p_1 \\ 0.000 \end{array}$$

Step-6, 7&8:

After implementing assignment process, the solution set, of the each of sub problem, the allocation is thus obtained.

Solution for the Sub Problem-I:

Tasks	→	Processors	ER
t_1	→	p_1	0.997
t_2	→	p_2	0.998
t_4	→	p_3	0.998

Solution for the Sub Problem-II:

Tasks	→	Processors	ER
t_3	→	p_1	0.998

Step-9:

After implementing the process, we obtain the following set of complete assignments along with execution reliabilities of each processor.

$$
\begin{array}{ccccc}
Tasks & \rightarrow & Pr\,ocessors & & ER \\
t_1 & \rightarrow & p_1 & & 0.997 \\
t_2 & \rightarrow & p_2 & & 0.998 \\
t_3 & \rightarrow & p_1 & & 0.998 \\
t_4 & \rightarrow & p_3 & & 0.998 \\
\end{array}
$$

Step-10:

ER	$:=$	0.99102990
CR	$:=$	0.96645590
Ereliability	$:=$	0.95778660

Step-11:
   Stop.

Example-II

Now consider a system which consists a set $T = \{t_1, t_2, t_3, t_4, t_5\}$ of 5 tasks and a set $P = \{p_1, p_2, p_3\}$ of 3 processors, where,

Step-1: Input: 5, 3

$$
ERM\,(,) =
\begin{array}{c|ccc}
 & p_1 & p_2 & p_3 \\
t_1 & 0.997 & 0.996 & 0.994 \\
t_2 & 0.993 & 0.998 & 0.992 \\
t_3 & 0.998 & 0.991 & 0.994 \\
t_4 & 0.997 & 0.993 & 0.998 \\
t_5 & 0.992 & 0.995 & 0.996 \\
\end{array}
$$

$$
CRM\,(,) =
\begin{array}{c|ccccc}
 & t_1 & t_2 & t_3 & t_4 & t_5 \\
t_1 & 0.000 & 0.994 & 0.996 & 0.995 & 0.993 \\
t_2 & 0.994 & 0.000 & 0.992 & 0.993 & 0.995 \\
t_3 & 0.996 & 0.992 & 0.000 & 0.992 & 0.996 \\
t_4 & 0.995 & 0.993 & 0.992 & 0.000 & 0.992 \\
t_5 & 0.993 & 0.995 & 0.996 & 0.992 & 0.000 \\
\end{array}
$$

Step-2:
   Obtain the product of each row and column of ERM (,), i .e. the products of each row and each column are as:

$$Product\_Row= \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \\ 0.9870539 & 0.9830858 & 0.9830838 & 0.9880409 & 0.9830918 \end{array}$$

$$Product\_Column = \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.9771962 & 0.9732756 & 0.9742587 \end{array}$$

Step-3:

We partitioned the matrix ERM (,) to define the first sub problem ERMI (,) by selecting rows corresponding to t_1, t_4, t_5 and second sub problem ERMII (,) by selecting rows corresponding to the tasks t_2, t_3 and by deleting columns corresponding to p_2. So that the modified matrices for each sub problems are as;

Sub Problem-I:

$$ERMI (,,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_1 & 0.997 & 0.996 & 0.994 \\ t_4 & 0.997 & 0.993 & 0.998 \\ t_5 & 0.992 & 0.995 & 0.996 \end{array}$$

and, Sub Problem-II:

$$ERMII(,,) = \begin{array}{c|cc} & p_1 & p_3 \\ t_2 & 0.993 & 0.992 \\ t_3 & 0.998 & 0.994 \end{array}$$

Obtain the unreliability matrices for the sub problems ERMI(,) and ERMII(,) as,

$$EURMI(,,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_1 & 0.003 & 0.004 & 0.006 \\ t_4 & 0.003 & 0.007 & 0.002 \\ t_5 & 0.008 & 0.005 & 0.004 \end{array}$$

$$EURMII(,,) = \begin{array}{c|cc} & p_1 & p_3 \\ t_2 & 0.007 & 0.008 \\ t_3 & 0.002 & 0.006 \end{array}$$

Step-4 & 5:

On applying modified Hungarian method devised by Kumar et al [KUMA 1995c] to assign the tasks, on the basis of min $\{r_i\}$ and min $\{c_j\}$ from unreliability matrices for every i and j. We put $r_{ij} = 0$ and $c_{ij} = 0$, for every i and j. On applying this to all sub problems, the modified matrices for each sub problem are mentioned below:

$$
EURMI(,,) = \begin{array}{c c c c} & p_1 & p_2 & p_3 \\ t_1 & 0.000 & 0.000 & 0.006 \\ t_2 & 0.003 & 0.007 & 0.000 \\ t_4 & 0.008 & 0.005 & 0.000 \end{array}
$$

$$
EURMII(,,) = \begin{array}{c c c} & p_1 & p_3 \\ t_2 & 0.000 & 0.008 \\ t_3 & 0.000 & 0.000 \end{array}
$$

Step-6, 7&8:

After implementing assignment process, the allocation is thus obtained.

Tasks	→	Processors	ER
t_1	→	p_2	0.996
t_2	→	p_1	0.993
t_3	→	p_3	0.994
t_4	→	p_1	0.997
t_5	→	p_3	0.996

Step-9:

ER   :=   0.9762239

CR   :=   0.9501149

Ereliability   :=   0.9275249

Step-10:

Stop.

3.5.5 Conclusion

The algorithm presented here is to be capable for maximizing the overall reliability of distributed system through task allocation. In distributed system tasks are allocated in such a way that their individual reliability of processing is optimized as well as it improve the overall reliability. In this approach not only that the loads of each processor get evenly balanced and none of the task gets unprocessed so that requirement for adding dummy processor became meaningless in our approach. Our approach provide an optimal solution for assigning a set of "m" tasks of a program to a set of "n" processors where m > n in a distributed system that to maximize the overall reliability of the system and the load of all allocated tasks on all the processors equally balanced. For the example-I, the execution reliability on different processors has been obtained. Also the communication reliability of the distributed system is mentioned in the following table:

Processors of the Distributed Systems	p_1	p_2	p_3
Execution reliability of each processor PER (,)	0.995006	0.998000	0.998000
Communication reliability of distributed system	0.96645590		
Total reliability of distributed system [Ereliability(,)]	0.95778660		

The final results of example-II are as:

Tasks	$\rightarrow$	Pr ocessors	ER	CR	Ereliabili ty
$t_1 * t_3$	$\rightarrow$	p_1			
t_2	$\rightarrow$	p_2	0.99102990	0.96645590	0.95778660
t_4	$\rightarrow$	p_3			

For the example-II, the execution reliability on different processors has been obtained. Also the communication reliability of the distributed system is mentioned in the following table:

Processors of the Distributed Systems	p_1	p_2	p_3
Execution reliability of each processor PER (,)	0.990024	0.996000	0.990021
Communication reliability of distributed system	0.9501149		
Total reliability of distributed system [Ereliability(,)]	0.9275249		

The final results of Example-II are as:

$$
\begin{array}{ccccc}
Tasks & \rightarrow & Processors & ER & CR & Ereliability \\
t_1 & \rightarrow & p_2 & & & \\
t_3*t_5 & \rightarrow & p_3 & 0.9762239 & 0.9501149 & 0.9275249 \\
t_2*t_4 & \rightarrow & p_1 & & & \\
\end{array}
$$

The method is presented in computational algorithmic form and implemented on the several sets of input data to test the performance and effectiveness of the algorithm. The model discussed here, would be useful to the network system designer working in the field of distributed systems. The developed model is programmed in C++ and implemented the several sets of input data to test the effectiveness and efficiency of the algorithm. It is recorded that the model is suitable for arbitrary number of processors with the random program structure.

FUTURE SCOPE AND PUBLICATIONS

4.1.   FUTURE SCOPE

The present research work entitled "APPLICATIONS OF MATHEMATICAL PROGRAMMING IN THE DEVELOPMENT OF SOME MODELS FOR PERFORMANCE EVALUATION OF DISTRIBUTED SYSTEMS" is devoted to the optimization techniques, distributed systems, task allocation and other directly related to the issues. Modeling completion time of algorithms is an interesting issue that needs to be investigated. In case of uniprocessor system, completion time is number of arithmetic operations required to complete the algorithm. Additional operations such as IPC, synchronization and global decisions making may add to the completion time of the algorithm on multiple computer system. Modeling completion time of an algorithm on an array processor system is presented by Adams [Adam 84]. This issue requires deep concern and can be investigated further. A key issue in distributed software engineering is partitioning an application into a set of tasks. It is worth exploring the possible techniques for partitioning so as to enhance the system performance. Some partitioning strategies for solutions of partial differential equation problems have been derived by Berger and Bokhari [BERG85, BOKH 87], and Siva Rama Murthy and Raja Raman [SIVA 88]. The design of a new product involves specified performance requirements, evaluation & selection of components to perform clearly defined functions, and

determination of a system level architecture. Detailed system engineering specifications prescribe reliability index. If the design is then to be produced economically or within some specified budget, various design alternatives must be considered. In the real life system, other issues must be taken into account, including memory restrictions, load balancing and queuing delays. How to integrate these various factors into a model of task allocation to make it more realistic and tractable remains a challenging problem to be discussed about. Task assignment of any distributed system is a most computing interesting and demandable research problem. Various methodologies and techniques are available in the literature to solve such problems. Keeping in view, the importance and necessity of performance evaluation of communication system, the future aim of study would be to develop some algorithms and techniques regarding the following problems.

- Developing new techniques & their algorithm for proper utilization of the processors of the distributed system to avoid overloading the processors.

- Developing the algorithm for assignment using artificial neural network.

- To devise some approaches to evaluation of the performance of distributed communication system.

- Developing the criterion for forming the clusters has to be taken into consideration and its implementation.

- Developing some of the dynamic algorithms for assignment policy and its implementation on various fields of daily life.

4.2. TECHNICAL PUBLICATIONS

1. A research paper entitled, "MATHEMATICAL PROGRAMMING APPROACH FOR ASSIGNING THE TASKS TO THE PROCESSORS IN A DISTRIBUTED ENVIRONMENT" Presented at 71st Annual Conference of IMS during 26-29 Dec. 2005 at IIT, Roorkee.

2. A research paper entitled, "A LOAD BALANCING APPROACH FOR OPTIMIZING THE RESOURCES OF DISTRIBUTED SYSTEMS" presented at the 12th Annual Conference of Gwalior Academy of mathematical Sciences (GAMS) during April 06-08, 2007 at Department of Mathematics, MANIT, Bhopal.

3. A research paper entitled, "MATHEMATICAL PROGRAMMING APPROACH FOR ASSIGNING THE TASKS TO THE PROCESSORS IN A DISTRIBUTED ENVIRONMENT" presented at the National Conference on Applications in Science, Technology & Management during 06-07 Sept. 2008 at SRMCET, Bareilly.

4. A research paper entitled, "RELIABILITY BASED STATIC APPROACH FOR PERFORMANCE ENHANCEMENT OF DISTRIBUTED SYSTEMS", communicated to Journal of IAPS.

REFERENCES

[AGAR 1995]   Aggarwal, D. C., "Computer Communication and ISDN Systems", Khanna Publishers, New Delhi, 1995.

[AGGA 1982]   Aggarwal, K.K., Chopra, Y. C., and Bajwa, J. S., "Reliability Evaluation by Network Decomposition", IEEE Transactions on Reliability, Vol. R-31, No. 4, 1982, pp. 355-358.

[ALFO 1988]   Alford, M.W., "Heuristic Algorithm for Task Assignment in Distributed Systems", IEEE Transaction on Computers, Vol. 37, No. 11, 1988, pp. 1384-1397.

[AMAR 1998]   Amari, S. V., " Reliability, Risk and Fault – Tolerance of Complex Systems", Ph. D. Dissertation, Indian Institute of Technology, Kharagpur, 1998.

[AROR 1979]   Arora, R. K., and Rana, S. P., "On Module Assignment in Two Processors Distributed Systems", Information Processing Letters, Vol. 9, No. 3, 1979, pp. 113-117.

[AROR 1980]   Arora, R. K., and Rana, S. P., " Heuristic Algorithms for Process Assignment in Distributed Computing Systems", Information Processing Letters, Vol. 11, No. 45, 1980, pp. 199-203.

[AROR 1981]   Arora, R. K., and Rana, S. P., "On the Design of Process Assigner for Distributed Computing Systems", The Australian Computer Journal, Vol. 13, No. 3, 1981, pp. 77-82.

[ASHR 1992]   Ashrafi, N., and Berman, O., "Optimization Models for Selection of Programs, Considering Cost and Reliability", IEEE Transactions on Reliability, Vol. 41, No. 2, 1992, pp. 281-287.

[BACA 1989]   Baca, D.F., "Allocation Modules to Processor in a Distributed Systems", IEEE Transactions on Software Engineering, Vol. SE-15, 1989, pp. 1427-1436.

[BERG 1985]   Berger, M. J., and Bokhari, S. H., " A Partitioning Strategy for PDEs Across Multiprocessors", Proc. of the 1985, International Conference on Parallel Processing, 1985, pp. 166-170.

[BERG 1987]   Berger, M. J., and Bokhari, S. H., "A Partitioning Strategy for Non Uniform Problems on Multiprocessors", IEEE Transactions on Computers, Vol. C-36, No. 5, 1987, pp. 570-580.

[BERM 1984]   Berman, F., and Synder, L., "On Mapping Parallel Algorithms into Parallel Architecture", Proceeding of the International Conference on Parallel Processing, 1984, pp. 307 - 309.

[BHAR 1994]   Bharadwaj, V., Ghosh, D., and Mani, V., "Optimal Sequencing and Arrangement in Distributed Single-Level Tree Network with Communication Delays", IEEE Transactions on Parallel & Distributed Systems, Vol.5, No. 9, 1994, PP. 968-976.

[BHUT 1994]   Bhutani, K.K., "Distributed Computing", The Indian Journal of Telecommunication, 1994, PP. 41-44.

[BIER 2002]   Bierbaum L. Rene, Brown D. Thomas, and Kerschen, J., Thomas., "Model Based Reliability Analysis", IEEE Transactions on Reliability, Vol. 51, No. 2, 2002, pp. 133-140.

[BOKH 1979]   Bokhari, S. H., "Dual Processor Scheduling with Dynamic Reassignment", IEEE Transactions on Software Engineering, Vol. SE-5, 1979, pp. 341-349.

[BOKH 1981a] Bokhari, S. H., " A Shortest Tree Algorithm for Optimal Assignment Across Space and Time in Distributed processor System", IEEE Transactions on Software Engineering, Vol. SE-7, No. 6, 1981, pp. 583-589.

[BOKH 1987]   Bokhari, S.H., "Assignment Problems in Parallel and Distributed Computing", Kluwer Academic Publisher, 1987.

[BOKH 1988]   Bokhari, S.H., "Partitioning Problems in Parallel, Pipeline and Distributed Computing", IEEE Transactions on Computers, Vol. C-37, No. 1, 1988, pp. 48-57.

[BOKH 1993]   Bokhari, S.H., "A Network Flow model for Load Balancing in Circuit-Switched Computer", IEEE Transactions on Parallel & Distributed Systems, Vol. 4, No. 6, 1993, PP. 649-657.

[BOLC 1983]   Bolch, G., Hofmann, F., Hoppe, B., Kolb, H.J., Linster, C.U., Polzer, R., Schussler, W., Wackersreuther, G., and Wurm, F.X., "A Multi Processor System for Simulating data Transmission System (MUPSI)", Micro-processing and Microprogramming, Vol. 12, 1983, pp. 267-277.

[BOWE 1992]   Bowen, N. S., Nikolaou, C. N., and Ghafoor, A., "On the Assignment Problem of Arbitrary Process Systems to Heterogeneous Distributed Computer Systems," IEEE Transactions on Computers, Vol. 41, No. 3, 1992.

[BUCK 1979]   Buckels, B.P., and hardin, D.M., "Partitioning and Allocation of Logical Resources in a Distributed Computing Environment', Tutorial: Distributed System Design", IEEE Computer Society Press, 1979.

[CASA 1988]   Casavant, T. L., and Kuhl, J. G., "A Taxonomy of Scheduling in General Purpose Distributed Computing System", IEEE Transactions on Software Engineering, Vol. SE-14, 1988, pp. 141-154.

[CHIN 2006]   Chin-Ching Chiu, Chung-Hsien Hsu, Yi-Shiung Yeh, "A Genetic Algorithm for Reliability-Oriented Task Assignment with k/spl tilde / duplication in Distributed Systems", IEEE Transactions on Reliability, Vol. 55, No. 1, 2006, PP. 105-117.

[CHOU 1986]   Chou, T.C.K., and Abraham, J.A., "Distributed Control of Computer Systems", IEEE Transactions on Computers, Vol. C-35, No. 6, 1986, PP. 564-567.

[CHU 1969]   Chu, W. W., "Optimal File Allocation in a Multiple Computing System", IEEE Transactions on Computers, Vol. C-18, 1969, pp. 885-889.

[CHU 1980]   Chu, W.W., Holloway, L. J., Lan, L. M. T., and Efe, K., "Task Allocation in Distributed Data Processing", IEEE Transactions on Computers, Vol. 13, No. 11, 1980, pp. 57-69.

[CHU 1980b]   Chu, W.W., "Introduction in the Special Issue on Distributed processing", IEEE Transactions on Computers, Vol. C-29, 1980, pp. 1037-1038.

[CHU 1984b]   Chu, W. W., and Leung, K.K., "Task Response Time Model and its Application for Real Time Distributed Processing Systems", Proceeding of 5th Real-Time Systems Symposium Texas, 1984, PP.225-236.

[CHU 1987b]   Chu, W. W., and Lan, L.M.T., "Task Allocation and Precedence Relation for Distributed Real-Time", IEEE Transactions on Computers, Vol. C-36, No.6, 1987, PP.667-679.

[CHUA 1992]   Chuang, Po-jen, and Tzeng, Nian Feng, " A Fast Recognition Complete Processor Allocation Strategy Hypercube Computer", IEEE Transactions on Computers, Vol. 41, No. 4, 1992, pp. 467-479.

[COIT 1996]   Coit, W., David, and Smith, A. E., "Reliability Optimization of Series-Parallel Systems using Genetic Algorithm", IEEE Transactions on Reliability, Vol. 45, No. 2, 1996, pp.254-266.

[COIT 1998a]   Coit, W., David, and Smith, A. E., "Redundancy Allocation to Maximize a Lower Percentile of the System Time-to-Failure Distribution", IEEE Transactions on Reliability, Vol. 47, No. 1, 1998, PP. 79-87.

[COIT 1998b]   Coit, D. W., "Economic Allocation of Test Times for Sub-System level Reliability Growth Testing," IEEE Transaction on Reliability, Vol. 30, No. 12, 1998, pp.1143-1151.

[COST 2007]   Costa and E.O. de Souza, G.A., Pozo, A.T.R., Vergilio, S.R., "Exploring Genetic Programming and Boosting Techniques to

Model Software Reliability," IEEE Transaction on Reliability, Vol. 56, No. 3, 2007, pp.422-434.

[DENG 1997]   Dengiz, Benna, Altiparmak, Fulya, and Smith Alice, E., "Efficient Optimization of All-Terminal Reliable Networks, Using and Evolutionary Approach", IEEE Transactions on Reliability, Vol. 46, No. 1, 1997, pp. 18-26.

[DENS 1998]   Denson, William, " The History of Reliability Prediction" IEEE Transactions on Reliability, Vol. 47, No. 3- sp, 1998, pp. SP-321-328.

[DESS 1980]   Dessoukiu-EI, O.I., and Huna, W.H., "Distributed Enumeration on Network Computers," IEEE Trans. On Computer, Vol.C-29 pp.818-825, 1980.

[DINI 1970]   Dinic, E.A., "Algorithm for Solution of a Program of Maximum Flowing a Network with Power Estimation", Soviet Math. Doklady, Vol. 11, No. 5, 1970, pp. 1277-1280.

[EDMO 1972]   Edmonds, J., and Karp, R. M., "Theoretical Improvements in Algorithms Deficiency for Network Flow Problems", J. Asso. Comp. Mech., Vol. 19, 1972, pp. 248-264

[EFE 1982]   Efe. K., " Heuristic Models of task Assignment Scheduling in Distributed Systems", IEEE Transactions on Computers, Vol. 5, No. 6, 1982, pp. 50-56.

[FABO 1976]   Fabozzi, F.J., and Valaente, J., "Mathematical Programming in American Companies : A sample survey", Interfaces, Vol. 7, 1976, PP. 93-98.

[FERN 1989]   Fernandez-Baca, David, "Allocating Modules to Processors in a Distributed Systems", IEEE Transactions of Software Engineering, Vol. SE-15, No. 11, 1989, pp. 1427-1436.

[FITZ 1988]   Fitzserald, J., "Business Data Communication", John Wiley & Sons, New York, 1988.

[FITZ 2002]   Fitzserald, Kent, Latifi, Shahram, Srimani, Pradeep, K., "Reliability Modeling and Assessment of the Star Graph Network", IEEE Transactions of Reliability, Vol. 51, No. 1, 2002, pp. 49-57.

[FORD 1962]   Ford, L. R. Jr., and Fulkerson, D. R., " Flows in Networks", Princeton University Press, 1962.

[FORT 1985]   Forter, P.J., "Design and Analysis of Distributed Real-Time System", Inter Text Publication Inc., and McGraw-Hill, Inc., New York, 1985.

[FUKU 1987]   Fukunage, K., Yamada, S., and Kasail, T., "Assignment of Job Modules on to Array Processors", IEEE Transactions on Computers, Vol. C-36, No. 7, 1987, pp. 881-891.

[GARC 1982]   Garcia-Molina, J., "Reliability Issues for Fully Replicated Database", IEEE Transactions on Computers, Vol. 16, 1982, PP. 34-42.

[GRAY 1973]   Grayson, C.J., "Management Science and Business Practice", Howard Business Review, Vol. 51, 1973, PP. 41-48.

[HA 2006]   Ha, C and Kuo, W.,"Multi-path Heuristic for Redundancy Allocation : The Three Heuristic", IEEE Transaction on Reliability, Vol. 55, No. 1, 2006, pp. 37-43.

[HOLT 1978]   Holt, A. W. et al., "Final Report of the Information System Theory Project", Room Air Development Center, No. AD, 1978, pp. 676- 972.

[HUI 1997]   Hui, C.C., and Chanson, S.T., "Allocation Task Interaction Graphs to Processors in Hetrogeneous Networks", IEEE Transactions on Parallel and Distributed System, Vol.8, No.9, 1997, PP. 908-915.

[HWAN 1993]   Hwang, G., and Teeng, S., "A Heuristic Task Assignment Algorithm to Maximize Reliability of Distributed System", IEEE Transactions on Reliability, Vol. 42, No. 3, 1993, pp. 408-415.

[IQBA 1986a]   Iqbal, M.A., "Approximate Algorithms for Partitioning and Assignment Problems", ICASE Report No. 86-40, 1986

[IQBA 1986b]   Iqbal, M.A., Saltz, J.H., and Bokhari, S.H., "A Comparative Analysis of Static and Dynamic Load Balancing Strategies", Proceeding of the 1986 International Conference on Parallel Processing, 1986, pp. 1040-1047.

[JOLL 1980]   Joller, J. M., "Reliability Block Diagrams and Petri Net", Microelectronics and Reliability, Vol. 20, 1980, pp.613-624.

[KARA 2008] Karasakal, Orhan, "Air Defense Missile-Target Allocation Model for a Naval Task Group", Published in Journal of Computer and Operation Research, vol. 35, 2008, pp.1759-70.

[KART 1995] Kartik, S., and Murthy, C. Siva Ram, "Improved Task-Allocation Algorithms to Maximize Reliability of Redundant Distributed Computing Systems", IEEE Transactions on Reliability, Vol. 44, No. 4, 1995, pp. 575-586.

[KARZ 1974] Karzanova, A. V., "Determining the Maximum Flow in Network by Method for Preflows", Soviet Math. Doklady, Vol. 15, No. 2, 1974, pp. 434-437.

[KE 1997] Ke, Wei-Jehn, and Wang, Sheng-De, "Reliability Evaluation for Distributed Computing Networks with Imperfect Nodes", IEEE Transactions on Reliability, Vol. 46, No. 3, 1997, pp. 342-349.

[KEIN 1971] Keingham, B.W., "Optimal Sequential Partition of Graphs", Journal of the ACM, Vol. 18, No. 1, 1971, PP. 34-40.

[KIM 1988] Kim, S. J. and Browne, " A General Approach to Mapping of Parallel Computations upon Multiprocessor Architectures", Proceeding International Conference on Parallel Processing, Vol. – 3, Aug 1998, pp. 1-8.

[KOHL 1975] Kohler, W.H.,"A Perliminary Evaluation of the Critical Path Method for Scheduling Tasks on Multiple Processor Systems," IEEE Trans. Computer, vol. 24, no. 12, pp. 1,235-1,238, Dec. 1975.

[KUMA 1988]   Kumar, Vinod, and Aggarwal, K.K., "Determination of Path Identifiers for Reliability Analysis of a Broadcasting Network Using Petri Nets", International Journal of Systems Sciences, Vol. 19, No. 12, 1988, pp. 2643-2653.

[KUMA 1990]   Kumar, Vinod, and Aggarwal, K.K., "Efficient Computerized Petri Net Approach for the Enumeration of the Sets of Path Identifier for Reliability Analysis of Broadcasting Network", International Journal of Systems Sciences, Vol. 21, No. 7, 1990, pp. 1239-1248.

[KUMA 1993]   Kumar, Vinod, and Aggarwal, K.K., "Petri Net Modeling and Reliability Evaluation of Distributed Processing Systems", Reliability Engineering and Systems Safety, Vol. 41, 1993, pp. 167-176.

[KUMA1995a]   Kumar, A., Pathak, R. M., Gupta, Y. P., and Parsaei H.R., "A Genetic Algorithm for Distributed System Topology Design", Computers and Industrial Engineering, Vol. 28, 1995, pp. 659-670.

[KUMA1995b]   Kumar, A., Pathak, R. M., Gupta, Y. P., "Genetic Algorithm based Reliability Optimization for Computer Network Expansion", IEEE Transactions on Reliability, Vol. 44, No. 1, 1995, pp. 63-72.

[KUMA 1995c]   Kumar, Vinod, Singh, M.P., and Yadav, P.K., "A fast algorithm for allocation task in distributed processing system,"

proceedings of the 30th Annual convention of computer society of India, Hyderabad, pp.347-358, 1995.

[KUMA 1995d]   Kumar, Vinod, Singh, M.P., and Yadav, P.K., "An efficient Algorithm for allocating tasks to processors in a distributed system," Proceedings of the Nineteenth National system conference (NSC-95), PSC College of technology, Coimbatore, organized by system society of India, New Delhi, pp. 82-87, 1995.

[KUMA 1996]   Kumar, Vinod., Singh, M.P., and Yadav, P.K.,"An Efficient Algorithm for Multi-processor Scheduling with Dynamic Re-Assignment", Proceeding of the 6th National Seminar on Theoretical Computer Science. Held at Banasthali Vidyapith, India, pp. 105-118, 1996.

[KUMA 1998]   Kumar, Vinod, Yadav, P.K., and Bhatia, K.,"Optimal Task Allocation in Distributed Computing Systems Owing to Inter Task Communication Effects." Proc. Of CSI-98 : IT for the New Generation, pp.369-378, 1998.

[KUMA 1999]   Kumar, Avanish, "Optimizing for the Dynamic Task Allocation", Published to the proceedings of the III Conference of the International Academy of Physical Sciences held at Allahabad, 1999, pp. 281-294.

[KUMA 2001]   Kumar, Avanish, "An Algorithm for Optimal Index to Task Allocation Based on Reliability and Cost", Published to the

proceedings of International Conference on Mathematical Moideling held at Roorkee, 2001, pp. 150-155.

[KUMA 2002]   Kumar, Avanish, and Yadav, P.K., "An Efficient Static Approach for Allocation Through Reliability Optimization in Distributed Systems", Presented at the International Conference on Operation Research for Development (ICORD2002) Held at Chennai.

[KUMA 2006]   Kumar, Harendra, Singh, M.P., Kumar, Avanish, "A Task Allocation Model for Distributed Data Network", Journal of Mathematical Sciences, Vol.1, no.4, 2006, pp.379-392.

[KUMA 2007]   Kumar, Harendra, Singh, M.P., Yadav, P. K., Kumar, Avanish, "An Efficient Tasks Assignment Algorithm in Hetrogeneous Distributed system", Varahmihir Journal of Computer & Information Sciences, Vol.2, no.1 &2, 2007, pp.105-116.

[KUMA2009]   Kumar, Subodha, Dutta, Kaushik, Mookerjee, Vijay, "Maximizing Business value by optimal assignment of jobs to resources in Grid Computing", Published in European Journal of Operation Research, Vol. 194, 2009, pp. 856-872.

[KWAK 1987]   Kwak, N.K., and Moor, J.S., "Introduction to Mathematical Programming", Robert E. Krieger Publishing Company Manabar, Florida. 1987.

[LAN 1985]   Lan, L. M. T., "Characterization of Inter Module Communications and Heuristic Task Allocation for Distributed

Real -Time System", Ph.D. Dissertation, Computer Science Department University of California, Los Angeles, 1985.

[LEDB 1977] Ledbetter, W.N., and Cox, J.F., "Are Or Techniques Being Used?", Industrial Engineering, Vol. 27, 1977, PP. 19-21.

[LEE 1987] Lee, S. Y., and Aggarwal, K.K., "A Mapping Strategy for Parallel Processing", IEEE Transactions on Computers, Vol. C-36, No. 4, 1987, pp. 433-441.

[LEVI 1998] Levitin, Gregory, Lisnianski, Anatoly, Ben-Haim, Hanoch, and Elmakis, David, "Redundancy Optimization for Series-Parallel Mult-State System", IEEE Transactions on Reliability, Vol. 47, No. 1, 1998, pp. 165-172.

[LI 1992] Li, Duan, and Haimes, Y. Y., " A Decomposition Method for Optimization of Large–system Reliability", IEEE Transactions on Reliability, Vol. 41, No. 2, 1992, pp. 183-188.

[LIAN 1994] Liang, Ting-Peng, Lai, H., Chen, Nain-Shing, Wai, H., and Chen, M. C., "When Client/Server Isn't enough: Coordinating Multiple Distributed Tasks", IEEE Computers, Vol. 27, No. 5, 1994, pp. 73-79.

[LIN 1990] Lin, F.T., and Hsu, C.C.," Task Assignment Scheduling by Simulated Annealing," IEEE Region 10 Conference on Computer and Communication Systems, pp.279-283, Hong Kong, September 1990.

[LIN 2002]   Lin, Min – Sheng, "A Linear Time Algorithm for Computing K-Terminal Reliability on Proper Interval Graph" IEEE Transactions on Reliability, Vol. 51, No. 1, 2002, pp. 58-62.

[LINT 1981]   Lint, B., and Agarwal, T., "Communication Issues in the Design and Analysis of Parallel Algorithm", IEEE Transactions on Software Engineering, Vol. SE-7, No. 2, 1981, pp. 174-188.

[LIU 1999]   Liu, Xiaoming, Kreitz, Christoph, Renesse, Robbert van, Hickey, Jason, Hayden, Mark, Kenneth, Birman, Constable, Robert, "Building reliable, high-performance communication systems from components", ACM Symposium on Operating Systems Principles, Charleston, South Carolina United States, 1999, pp. 80-92.

[LO 1988]   Lo, V.M., "Heuristic Algorithms for Task Assignments in Distributed Systems", IEEE Transactions on Computers, Vol. 37, No. 11, 1988, pp. 1384-1397.

[LOPE 1994]   Lopez-Benitez, N., "Dependability Modeling and Analysis of Distributed Programs", IEEE Transactions on Software Engineering, Vol. 20, No. 5, 1994, pp. 345-352.

[LU 1986]   Lu, H., and Carey, M. J., "Load Balanced Task Allocation in Locally Distributed Computer Systems", Proceeding of the 1986 International Conference on Parallel Processing, 1986, pp. 1037-1039.

[LYU 2002]   Lyu, Michael, R., Rangarajan, Sampath and Moorsel, and P.A. Van, "Optimal Allocation of Test Resources for Software

Reliability Growth Modeling in Software Development", IEEE Transactions on Reliability, Vol. R-51, 2002, pp. 183-192.

[MA 1982]   Ma, P. Y. R., Lee, E. Y. S., and Tuschiya, M., "A Task Allocation Model for Distributed Computing Systems", IEEE Transactions on Computers, Vol. C-31, No. 1, 1982, pp. 41-47.

[MANI 1998]   Manimaran, G., Murthy, C.S.R., "A Fault – Tolerant Dynamic Scheduling Algorithm for Multi-processor Real – time Systems and its Analysis," IEEE Transactions on Parallel and Distributed Systems, Vol. 9, No. 11, Nov. 1998, pp. 1137-1152.

[MARC 1981]   Marcogliese, R., and Novarese, R., "Module and Data Allocation Methods in Distributed Systems", Proceeding of the 1981 International Conference on Distributed Computer Systems, 1981, pp. 50-59.

[MARK 1972]   Markland, R.E., and Newett, R.J., "A Subjective Taxonomy for Evaluation the Stages of Management Science", Interfaces, Vol.2, 1972, PP. 31-39.

[MARK 1986]   Markenscoff, P., and Liew, W., "Task Allocation Problems in Distributed Computer System", Proceeding of the 1986 International Conference on Parallel Processing, 1986, PP. 953-960.

[MART 1988]   Martin, J., "Computer Network and Distributed Processing (Software Techniques and Architecture)", Prentice Hall of India Pvt. Ltd., Delhi, 1988.

[MEND 1987]   Mendelson, B., and Silberman, G.M., "An Improved Mapping of Data Flow Programs on a VLSI Array of Processor", Proceeding of the 1987 International Conference on Parallel Processing, 1987, pp. 871-874.

[MURA 1979]   Murata, T., "Relevance of Network Theory to Models of Distributed Processing", Proceeding 1979 International Colloquium Circuits systems, Japan, 1979.

[MURT 1988]   Murthy, C., Siva Ram, and Rajaraman, V., "Multiprocessor Architecture for Solving PDE's", Journal of the Institution of Electronics and Telecommunication Engineers, Vol. 34, No. 3, 1988, pp. 172-184.

[MURT 1989]   Murthy, C., Siva Ram, and Rajaraman, V., "Task Assignment in a Multiprocessor System", Micro Processing and Microprogramming, Vol. 26, 1989, pp. 63-71.

[NAKA 1976]   Nakazawa, H., "Bayesian Decomposition Method for Computing the Reliability of an Oriented Network", IEEE Transactions on Reliability, Vol. R-25, No. 2, 1976, pp. 77-80.

[NAKA 1977]   Nakazawa, H., " A Decomposition Method for Computing System Reliability by a Boolean Expression", IEEE Transactions on Reliability, Vol. R-26, No. 4,1977, pp. 250-252.

[NAKA 1978]   Nakazawa, H., " A Decomposition Method for Computing System Reliability by a Matrix Expression", IEEE Transactions on Reliability, Vol. R-27, No. 5, 1978, pp. 342-344.

[NAKA 1981]   Nakazawa, H., "Decomposition Method for Computing the Reliability of Complex Networks", IEEE Transactions on Reliability, Vol. R-30, No. 3, 1981, pp.289-292.

[OLSO 1994]   Olson, A., and Shin, K. G., " Fault-Tolerant Routing in Mesh Architectures", IEEE Transactions on Parallel and Distributed Systems, Vol. 5, No. 11, 1994, pp. 1225-1232.

[ONIS 2007]   Onishi J., Kimura, S., James, R.J.W., Nakagawa, Y., "Solving the Redundancy Allocation Problem with a Mix of Components using the Improved Surrogate Constraint Method, " IEEE Transaction on Reliability, Vol. 56, No. 1, 2007, pp. 94-101.

[PAIN 1995]   Painton, L., and Compbell, J., "Genetic Algorithms in Optimization of System Reliability", IEEE Transactions on Reliability, Vol. 44, No. 2, 1995, pp. 172-178.

[PENG 1997]   Peng, Dar-Tezen, Shin, K.G., and Abdel Zoher T.F., "Assignment Scheduling Communication Periodic Task in Distributed Real Time System", IEEE Transactions on Software Engineering, Vol. SE-13, 1997, pp. 745-757.

[PETR 1962]   Petri, C.A., "Communication with Automata", Transactions C. P. Greene, Jr, (Kommunikation Mit Automatcn), University of Bonn, 1962.

[PRIC 1982]   Price, C.C., and Pooch, U.W., "Search Techniques for Non-Linear Multiprocessor Scheduling Problems", Naval Research Logistics Quarterly, Vol. 29, No. 2, 1982, PP. 213-233.

[PRIC 1984]   Price, C.C., and Krishnaprasad, S., "Software Allocation Models for Distributed Computing Systems", Proceeding of the 4th International Conference on Distributed Computing Systems, 1984, pp. 40-48.

[RADN 1973]   Radnor, M., and Neal, R.D., "The Process of Management Science Activities in Large U.S. Industrial Corporation", Operation Research, Vol. 21, 1973, PP. 427-450.

[RAGH 1985]   Raghavendra, C. S., and Hariri, S., "Reliability Optimization in the Design of Distributed Systems", IEEE Transactions on Software Engineering, Vol. SE-11, 1985, pp. 1184-1193.

[RAI 1982]   Rai, Suresh, "A Cut- Set Approach to Reliability Evaluation in Communication Networks", IEEE Transactions on Reliability, Vol. R-31, No. 5, 1982, pp. 428-431.

[RAMI 2006]   Ramirez-Marquiz, J.E. and Wei, Jian.,"On Improved Confidence Bounds for System Reliability," IEEE Transaction on Reliability, Vol. 55, No. 1, 2006, pp. 26-36.

[RAMI 2007]   Ramirez–Marquez, J.E. and Gebre, B.A., "A Classification Tree Based Approach for the Development of Minimal Cut and Path Vectors of a Capacitated Network," IEEE Transactions on Reliability, Vol. 56, No. 3, 2007, pp. 474-487.

[REEV 1993]   Reeves, C. R., "Modern Heuristic Techniques for Combinational Problems," John Wiley & Sons, 1993.

[REID 1994]   Reid, D. J., "Optimal Distributed Execution of Join Queries", Computer Math. Application, Vol. 27, No. 11, 1994, pp. 27-40.

[RICH 1982] Richard, R.Y., Lee, E.Y.S., and Tsuchiya, M., "A Task Allocation Model fro Distributed Computer System", IEEE Transactions on Computers, Vol. C-31, 1982, pp. 41-47.

[ROSE 1982] Rosenthal, A., "Dynamic Programming is Optimal for Non Serial Optimization Problems", SIAM, journal of Computer, Vol. 11, No. 1, 1982, pp. 47-59.

[SAGA 1991] Sagar, G., and Sarje, A. K., "Task Allocation Model for Distributed System", International Journal of Systems Sciences, Vol. 22, No. 9, 1991, pp. 1671-1678.

[SCHN 2007] Schneeweiss, W.G., "Review of Petri Net Picture Book" and "Petri Nets for Reliability Modeling.", IEEE Transaction on Reliability, Vol. 55, No. 2, 2006, pp. 391-392.

[SEGE 1994] Segee, B.E., and Carter, M.J., "Comparative Fault Tolerance of parallel Distributed Processing networks", IEEE Trans. on Computer, Vol. 43, No.11, 1994, pp. 1323-1329.

[SHAT 1987] Shatz, S.M., and Wang, Jai-Ping, "Introduction to Distributed Software Engineering", Computers, Vol. 20, No. 10, 1987, pp. 23-30.

[SHAT 1989] Shatz, S.M., and Wang, Jai-Ping, " Models and Algorithms for Reliability-Oriented Task, Allocation in Redundant Distributed Computer Systems", IEEE Transactions on Reliability, Vol. 38, No. 1, 1989, pp. 16-27.

[SHAT 1992] Shatz, S. M., and Wang, Jai-Ping, " Task Allocation for Maximizing Reliability of Distributed Computer Systems",

IEEE Transactions on Computers, Vol. 41, No. 9, 1992, pp. 1156-1168.

[SHMI 1991]   Shmilovici, A., and Maimon, O. Z., "Estimating the Performance of Multi-process, Distributed Processing System that is Controlled with Fixed Priorities", 17th Convention of Electrical and Electronics Engineers in Israel (Proceedings), No. 90TH0360-8, 1991, pp. 408-409.

[SHOO 1983]   Shooman, M.L., "Software Engineering", Mc Graw - Hill Book Company, 1983.

[SIH 1993a]   Sih, G.C., and Lee, E.A., "A compile-time scheduling Heuristic for Interconnection-Constrained Heterogeneous processor Architectures," IEEE Trans. Parallel and Distributed systems, vol. 4. No. 2, pp. 175-186, Feb. 1993.

[SIH 1993b]   Sih, G. C., and Lee, E. A., "De-clustering: A New Multiprocessor Scheduling Technique", IEEE Transactions on Parallel and Distributed Systems, Vol. 4, No. 6, June 1993, pp. 625-637.

[SINC 1987]   Sinclair, J. B., "Efficient Computation of Optimal Assignment for Distributed Task", Journal of Parallel and Distributed Computing, Vol. 4, 1987, pp. 342-362.

[SING 1999]   Singh, M.P., Kumar, V., and Kumar, A., "An Efficient Algorithm for Optimizing Reliability Index in Task-Allocation", Acta Ciencia Indica, Vol xxv m, 1999, pp. 437-444.

[SINH 2004]   Sinha, Pradeep, K., "Distributed Operating Systems Concepts and Design", PHI, New Delhi, 2004.

[SITA 1995]   Sitaram, B.R., "Distributed Computing – A User's View Point", CSI Communications, Vol. 18, No.10, 1995, pp. 26-28.

[SIVA 1988]   Siva Ram Murthy, C., and Rajaraman, V., "Multiprocessor Architectures for Solving PDEs", Journals of the Institution of Electronics and Telecommunication Engineers, Vol. 34, No.3 1988, pp. 172-184.

[SIVA 1989]   Siva Ram Murthy, C., and Rajaraman, V., "Task Assignment in a Multiprocessor System", Micro processing and Microprogramming, Vol. 26, 1989, pp. 63-71.

[SRIN 1999]   Srinivasan, S., and Jha, N. K. "Safety and Reliability driven Task Allocation in Distributed Systems," IEEE Transactions on Parallel and Distributed Systems, Vol. 10, No. 3, Mar. 1999, pp. 238-251.

[STON 1977]   Stone, H. S., "Multiprocessor Scheduling with the Aid of Network Flow Algorithms", IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, 1977, pp.85-93.

[STON 1987]   Stone, H. S., (ed), "Introduction to Computer Architecture", Galgotia Publication Pvt. Ltd., New Delhi, 1987.

[TOWS 1986]   Towsley, D., "Allocation Program Containing Branches and Loop within a Multiprocessor System", IEEE Trans. on Software Engineering, Vol. SE-12, No. 10, 1986, PP. 1018-1024.

[TURB 1972]   Turban, E., "A sample survey of Operation Research Activities at the Corporate Level", Operation Research, Vol.20, 1072, PP. 708-721.

[WANG 2007]   Wang, N, Lu, J. C., Kvam, P., "Reliability Modeling in Spatially Distributed Logistics Systems", IEEE Transactions on Reliability, Vol. 55, No. 3, 2006, pp. 525-534.

[WARD 1984]   Ward, M. O., and Romero, D. J., "Assigning Parallel Executable Inter Communicating Subtasks to Processors", Proceeding of the 1984 International Conference on Parallel Processing, Bellaire, MI, 1984, pp. 392-394.

[WILL 1983]   Williams, E., "Assigning Processes to Processors in Distributed Systems", Proceeding of the 1983 International Conference on Parallel Processing, 1983, pp. 404-406.

[WU 1980a]   Wu, S. B., and Liu, M. T., "Assignment of Task and Resources for Distributed Processing", IEEE COMPCON fall 1980 Proceedings on Distributed Processing, 1980, pp. 665-672.

[WU 1980b]   Wu, S. B., and Liu, M. T., "A Partition Algorithm for Parallel and Distributed Processing", Proc. of the 1980 International Conference on Parallel Processing, 1980, pp. 254-255.

[YADA 2002]   Yadav, P.K., Singh, M.P. and Kumar, Harendra, " Task Allocation : An Algorithm for Systematic Allocation of tasks in Distributed computing Environment" in :National Seminar on Current trends in Mathematics and computation held at

Birla Institute of Technology, Ext. Center, NOIDA on December 1, 2005.

[YADA 2003] Yadav P. K., Kumar, Avanish and Sharma Manisha "An Optimal Sequencing Strategy for Module Execution in Computer Communication System" published in Journal of Mathematical Science, Vol. 3, pp. 175- 203.

[YADA 2008] Yadav P. K., Singh, M. P., Kumar, H., "Scheduling Algorithm: Tasks S cheduling Algorithm fo r Multiple Processors with Dynamic Reassignment" published in Journal of Computer Systems, Networks, and Communications, (Article in Press) 2008, Article ID 578180.

[YANG 1987] Yang, X. L., and Zhang, X. D., "A General Heuristic Algorithm of Task Allocation in Distributed Systems", Proc. of the 2nd International Conference on Computers and Applications, China, 1987, pp. 689-693.

[YANG 2008] Yang Bo, Huajun, Hu, Suchang, Guo, "Cost-Oriented task allocation and hardware Redundancy policies in heterogeneous distributed Computing Systems considering Software Reliability"', Published in Journal of Computer & Industrial Engineering (Article in Press) 2008, www.elsevier.com/locate/caie.

[YEN 2007] Yen, Wei Chang, "A Simple Heuristic Algorithm for Generating all Minimal Paths," IEEE Transactions on Reliability, Vol. 56, No. 3, 2007, pp. 488-494.

[YUAN 2006]   Yuan – Shun, Dai, Levitin, G., "Reliability and Performance of Tree Structured Grid Services", IEEE Transaction on Reliability, Vol. 55, No. 2, 2006, pp. 337-349.

[YUAN 2007]   Yuan – Shun, Dai, Levitin, G., "Optimal Source Allocation for Maximizing Performance and Reliability in Tree Structured Grid Services", IEEE Transaction on Reliability, Vol. 56, No. 3, 2007, pp. 444-453.

[ZAHE 1991]   Zahedi, F., and Ashrafi, N., "Software Reliability Allocation Based on Structure, Utility, Price, and Cost", IEEE Transactions on Software Engineering, Vol. 17, No. 4, April 1991, pp. 345-356.

[ZHOU 2007]   Zhou, Zhibo, Zhou, Tong and Jinxiang Wang., "Performance of Multi-User DCSK Communication System Over Multipath Fading Channels, " IEEE Transactions on Communications, Vol. 55, No. 9, 2007, pp. 2478-2481.